EWERTON CESÁRIO PERES

RPWNT - PROTOCOLO PARA COMUNICAÇÃO DE RÁDIO PACOTE: EXPERIMENTO NO AMBIENTE DE REDE WINDOWS NT

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Curso de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador:
Prof. Carlos Alberto Picanço de Carvalho

CURITIBA
2000

UFPR
Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Comissão Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno Ewerton Cesário Peres, avaliamos o trabalho intitulado **"RPWNT: Uma Especificação de Protocolo de Comunicação de Rádio Pacote para o Ambiente de Rede Windows NT"**, cuja defesa foi realizada no dia 12 de maio de 2000. Após a Avaliação, decidimos pela Aprovação do Candidato.

Curitiba, 12 de maio de 2000.

Prof. Dr. Carlos Alberto Picanço de Carvalho
Presidente - DINF/UFPR

Keiko V. O. Fonseca

Profª. Dra. Keiko Verônica Ono Fonseca
Membro Externo - CEFET-PR/ CPGEI

Elias Procópio Duarte Júnior

Prof. Dr. Elias Procópio Duarte Júnior
Membro - DINF/UFPR

"Se caráter custa caro, eu pago o preço."

Sidônio Muralha.

Dedico este trabalho aos meus pais Cesário e Ayda, que desde cedo me incentivaram ao estudo com responsabilidade, e à minha querida esposa Edilza pelo seu total apoio e companheirismo, estimulando-me e tendo paciência nos momentos de ausência e dificuldades.

AGRADECIMENTOS

Ao corpo docente do curso de mestrado em informática da renomada Universidade Federal do Paraná, pela dedicação durante minha formação. Em especial à Professora Doutora Olga Regina Pereira Bellon, pelo seu ótimo trabalho como coordenadora do curso de mestrado, e ao secretário José Carlos Pereira por ser sempre prestativo.

Ao meu orientador Professor Doutor Carlos Alberto Picanço de Carvalho, pela sua confiança, paciência e estímulo para me tornar um pesquisador.

Ao Professor Doutor Elias Procópio Duarte Murta pelo incentivo dado na fase final da dissertação e por aceitar fazer parte da banca examinadora, e à Professora Doutora Keiko Verônica Ono Fonseca, pela colaboração e presteza em também participar desta banca.

Aos queridos amigos e irmãos Letícia, Aldri e Fabiano Silva pelo apoio e incentivo, auxiliando-me tanto nas dúvidas quanto nas dificuldades.

Aos meus irmãos Aida, Jefferson, Letícia e Emerson, que sempre foram um ponto de apoio e busca de forças nos momentos difíceis.

Aos meus cunhados Fabiano Kuss, Ivane, Fabiano Silva, Jandi, Cláudio e Patrícia, bem como aos sobrinhos Alexandre, Thiaguinho, Emmanuel, Lucas, Lívia, Thiago, Fabrício e Bruno, pelas alegrias proporcionadas durante o transcorrer do curso e aos meus sogros Rosilda e Pedro, pelo carinho que sempre encontro.

Aos amigos Denilson, Jensen e Sérgio que, mesmo sem tempo, estiveram presentes com suas amizades.

À Companhia Paranaense de Energia - COPEL, através dos gerentes Leonardo Castilho e Robson Guarnieri, pela viabilização desta oportunidade de aprendizado.

Aos colegas da Informática da Câmara Municipal de Curitiba pelo apoio e solidariedade.

A todos que acreditaram em mim.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

ACK – *Acknowlegment*

API – *Application Program Interface*

AMPS – *Advanced Mobile Phone Service*

AX.25 – *Amateur X.25*

BIO – *Buffered Input/Output*

CDMA – *Code-Division Multiple Access*

CPDP – *Cellular Digital Packet Data*

CSMA/CA – *Carrier Sense with Multiple Access /Collision Avoidance*

CSMA/CD – *Carrier Sense with Multiple Access /Collision Detection*

CPU – *Central Process Unit*

DIFS – *Distributed Interframe Spacing*

DIO – *Direct Input/Output*

DISC – *Disconnect*

DLL – *Dynamic Link Library*

DM – *Disconnect Mode*

FAT – *File Allocation Table*

FCS – *Frame Check Sequence*

HAL – *Hardware Abstraction Layer*

HDLC – *High-Level Data Link Control Protocol*

HPFS – *High Performance File System*

HF – *High Frequency*

IEEE – *Institute of Electrical and Electronic Enginners*

I/O – *Input/Output*

IOCTL – *Input/Output Controls*

IPX/SPX – *Internet Packet Exchange / Sequenced Packet Exchange*

IRP – *Input/Output Request Packet*

IRQL – *Interrupt Request Level*

ISO – *International Standard Organization*

LAN – *Local Area Network*

LPC – *Local Procedure Call*

MAC – *Media Access Control*

MSAFD – Provedor de Serviços de Soquetes

MSDN – *Microsoft Developer Network*

NCPA – *Network Control Panel Application*

NDIS – *Network Driver Interface Specification*

NIC –*Network Interface Card*

NT – *New Tecnology*

NTFS – *NT File System*

OSI – *Open Systems Interconnections*

REJ – *Reject*

RNR – *Receive Not Ready*

RPWNT – Rádio Pacote para Windows NT.

RPC – *Remote Procedure Call*

RR – *Receive Ready*

SABM – *Set Async Balanced Mode*

SABME – *Set Async Balanced Mode Extended*

SMB – *Server Messages Blocks*

SREJ – *Selective Reject*

ST – *Sample Transport*

TCP/IP – *Transmission Control Protocol / Internet Protocol*

TDI – *Transport Driver Interface*

TDM – *Time-Division Multiplex*

TDMA – *Time-Division Multiple Access*

TNC – *Terminal Node Controller*

UA – *Unnumbered Acknowledge*

UI – *Unnumbered Information*

WAN – *Wide Area Network*

WIN32 – Biblioteca Windows 32 bits

WSH – *Windows Sockets Helper*

XID – *Exchange Identification*

RESUMO

A comunicação de dados sobre rádio pacote é uma das tecnologias utilizadas em ambientes de rede de computadores sem fio. A utilização da comunicação de rádio pacote nas últimas duas décadas se deve à interligação de milhares de estações de rádio amador formando uma rede global.

O protocolo AX.25 (*Amateur* X.25) é considerado um padrão mundial pela comunidade de rádio amador para utilização em rádio pacote, e atualmente implementado para os sistemas operacionais Unix, Linux e DOS. Apesar do sistema operacional Windows NT ter grande utilização em meios corporativos, não existe uma solução de comunicação de rádio pacote para este ambiente. Umas das razões é a alta complexidade de implementação de drivers na arquitetura de rede do Windows NT.

Este trabalho descreve a especificação de um protocolo baseado no protocolo de comunicação AX.25, como uma alternativa de comunicação de rádio pacote para o ambiente de rede Windows NT.

A especificação deste protocolo, chamado de RPWNT, possui características como a garantia de entrega dos pacotes através do controle de fluxo, o tratamento de erros, a independência do hardware utilizado e a capacidade de utilização de estações de rádio como repetidoras de pacotes.

Um protótipo foi implementado com a finalidade de observar a aplicabilidade do protocolo RPWNT ao ambiente de rede Windows NT. Este protótipo consiste no desenvolvimento de uma aplicação de usuário similar ao programa "ping" e dois componentes de rede, denominados biblioteca auxiliadora de soquetes e driver de transporte.

Para comprovar o funcionamento do protótipo, cada componente desenvolvido e a interligação entre eles foram testados. O protótipo conseguiu trocar mensagens com sucesso entre dois computadores ligados em rede. Além disto, a implementação permitiu explorar a arquitetura de rede deste sistema operacional e detectar problemas e dificuldades.

ABSTRACT

Packet radio data communications is one of the technologies employed by wireless computer networks. The utilization of packet radio communication in the last two decades is due to the fact that millions of amateur radio stations are already forming a global network.

The protocol AX.25 (*Amateur* X.25) is considered as world wide standard by the amateur radio community to the utilization of packet radio, and currently it is implemented in operating systems Unix, Linux and DOS. In spite of the fact that the Windows NT operation system has great utilization in corporative environment, there isn't a solution of packet radio communication to this operating system. One of the reasons is the complexity of drivers implementation to Windows NT network architecture.

This work describes the specification of a protocol based on AX.25 protocol communication, as an alternative of packet radio communication to the Windows NT network environment.

This protocol specification, which is called RPWNT, has characteristics as the warranty of packets delivery through the flow control, error treatment, independence of hardware and the usage of radio stations as packet repeaters.

A prototype was implemented with the aim to observe RPWNT's adaptability in the Windows NT network environment. This prototype consists of the development of a user application similar to "ping" and two network components, denominated windows sockets helper and transport driver.

The prototype was fully tested. Eeach developed component and the interrelation among them were tested. The prototype succeeded in exchanging messages between two computers on the network. Besides that, the implementation allowed us exploring this operating system network architecture and to detect problems and difficulties in implementing.

CAPÍTULO 1

INTRODUÇÃO

A interligação de milhares de estações de rádio amador ocorrida nas últimas décadas proporcionou o aparecimento de uma rede global de rádio pacote. Atualmente esta rede permite a comunicação entre computadores do mundo inteiro que utilizam os sistemas operacionais Unix, Linux e DOS.

Entre os vários protocolos para comunicação de rádio pacote, o protocolo AX.25 (*Amateur* X.25) é utilizado pela comunidade de rádio amador como padrão mundial. As características que o levaram a ser adotado como padrão são: a garantia de entrega dos pacotes através do controle de fluxo, o tratamento de erros e a capacidade de utilização de estações de rádio como repetidoras de pacotes, que não encontradas em outros protocolos.

O sistema operacional Windows NT, que tem grande utilização em meios corporativos, não possui uma solução de comunicação para ambiente de rádio pacote. Isto se deve ao fato do Windows NT possuir uma arquitetura proprietária altamente complexa que dificulta a implementação de qualquer tipo de driver.

Este trabalho propõe a especificação de um protocolo de comunicação de rádio pacote para o ambiente de rede Windows NT, denominado RPWNT, que é baseado na especificação do protocolo AX.25, versão 2.2 de 1997, de Terry L. Fox [BN97].

A simples implementação do protocolo AX.25 no ambiente de rede Windows NT o tornaria dependente da adaptadora de rede, pois segundo a sua especificação este protocolo deve trabalhar nas camadas físicas e enlace. Assim, para cada diferente adaptadora de rede, o protocolo teria que ser incluído no driver do dispositivo da adaptadora, que é a parte do código que trata particularidades do hardware. Já a especificação do protocolo RPWNT, por trabalhar nas camadas de rede e transporte, é portável para diferentes adaptadoras de rede.

Com a finalidade de observar a aplicabilidade do protocolo RPWNT ao ambiente de rede Windows NT, é implementado um protótipo que troca mensagens com sucesso entre dois computadores ligados em rede, simulando um "ping".

O texto está organizado em seis capítulos. O capítulo 2 apresenta a arquitetura interna de rede do Windows NT, fazendo um comparativo entre seus componentes e as camadas do modelo OSI. O protocolo proposto é contextualizado dentro desta arquitetura e é mostrado porque é portável para diversos meios de transmissão, sendo independente do hardware utilizado.

O capítulo 3 apresenta o estudo de alguns protocolos e tecnologias de comunicação em redes sem fio, com a finalidade de observar as características e procedimentos para este tipo de comunicação. Suas vantagens e desvantagens em relação ao uso em ambiente de rádio pacote são mencionadas. Os trabalhos relacionados que serviram como base para este estudo de protocolos e tecnologias são mencionados.

O capítulo 4 descreve a especificação do protocolo de transporte RPWNT para comunicação de rádio pacote a ser utilizado no ambiente de rede Windows NT, baseado no protocolo AX.25.

O capítulo 5 apresenta um protótipo do protocolo proposto com o objetivo de verificar a sua adaptabilidade no ambiente Windows NT, seguindo os padrões da arquitetura de rede deste sistema operacional. A implementação deste protótipo permitiu a observação e discussão da experiência, bem como a detecção de problemas e dificuldades desta adaptação.

No capítulo 6 são apresentadas as conclusões finais deste trabalho, com suas perspectivas de continuidade.

CAPÍTULO 2

ARQUITETURA DE REDE DO WINDOWS NT

Este capítulo tem o objetivo de contextualizar o protocolo de transporte proposto dentro do ambiente de rede Windows NT. Para tal, são apresentados os componentes internos da arquitetura de rede deste sistema operacional. A seção 2.1 descreve os componentes de rede do Windows NT, a seção 2.2 apresenta as camadas limítrofes entre estes componentes e a seção 2.3 detalha o uso de soquetes neste ambiente.

O Windows NT é um sistema operacional de 32 bits para redes de computadores, possuindo características como multitarefa, multithread, multiprocessamento simétrico e gerência de memória virtual. Além dessas características, o Windows NT também oferece suporte a múltiplas plataformas, a serviços de acesso remoto, a múltiplos protocolos e clientes e a múltiplos sistemas de arquivos [MM96], sendo muito utilizado em ambientes corporativos.

Os serviços de comunicação do Windows NT são separados em componentes que possuem funções específicas que não interferem nos trabalhos e resultados dos outros componentes. Esta arquitetura garante ao Windows NT uma facilidade muito grande de utilizar variados tipos de sistemas de arquivos, serviços de rede, protocolos de comunicação e dispositivos físicos.

Os componentes de rede do Windows NT são organizados em três principais categorias: drivers de sistemas de arquivos, drivers de transporte, e drivers do dispositivo. A figura 2.1 ilustra estes componentes, na coluna "Componentes do Windows NT", em comparação com as camadas do modelo OSI [Tan94] da *International Standard Organization* (ISO), representadas na coluna "Camadas OSI".

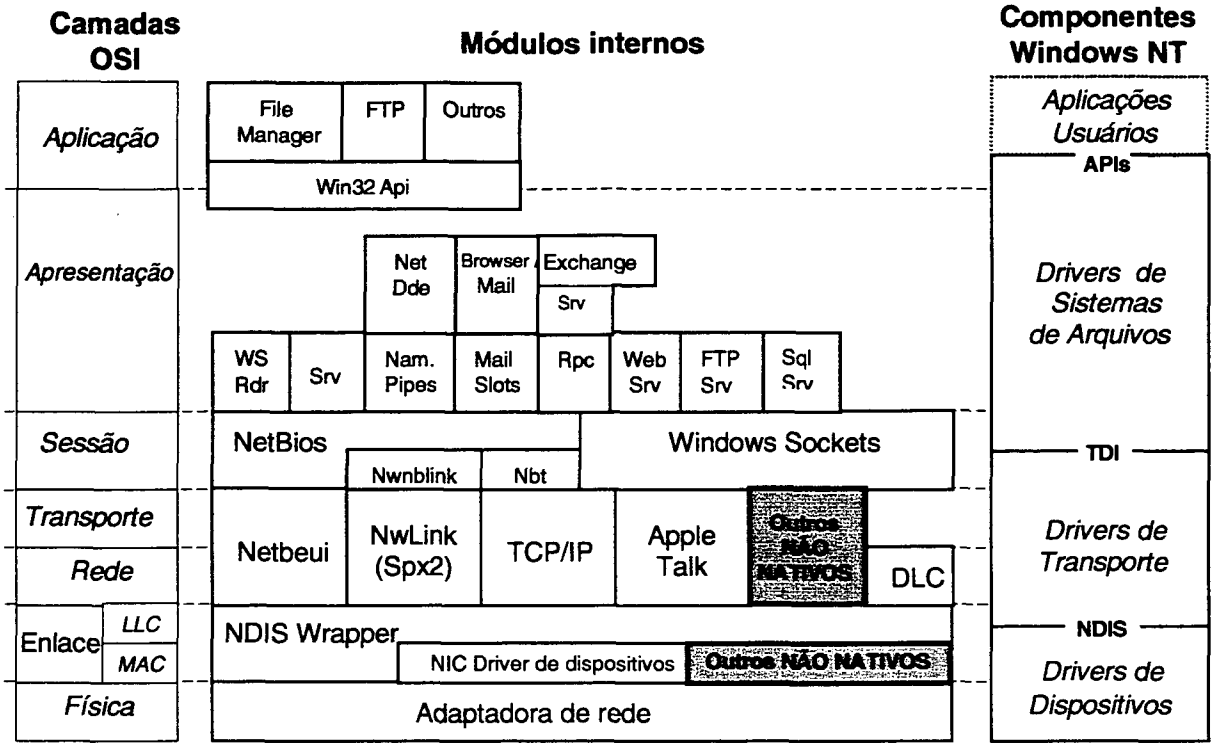


Figura 2.1 – Comparação da arquitetura de rede do Windows NT com o modelo OSI, baseado em [Stu95].

A coluna "Módulos internos" apresenta alguns módulos que são utilizados na comunicação em rede do Windows NT. A apresentação destes módulos é apenas ilustrativa, não fazendo parte do contexto deste trabalho. Mais detalhes de suas funcionalidades podem ser encontrados em [Sdk96, MH94, Stu95].

Os componentes da arquitetura de rede do Windows NT são independentes, podendo ser substituídos por outros em dois computadores que trocam informações, e mesmo assim a comunicação continuar sendo realizada com sucesso.

Cada componente do ambiente de rede Windows NT se comunica com seus subseqüentes através de bibliotecas de rotinas conhecidas como *camadas limítrofes*. No caso dos drivers de transporte, estes usam a camada limítrofe NDIS (*Network Driver Interface Specification*) para se comunicarem com o driver do dispositivo e usam a camada limítrofe TDI (*Transport Driver Interface*) para se comunicarem com os drivers de sistemas de arquivos. As seções seguintes apresentam cada componente do ambiente de rede Windows NT e as camadas limítrofes introduzidas anteriormente.

2.1 Drivers

No primeiro nível de componentes do Windows NT, estão os drivers de sistemas de arquivos que operam nas camadas de aplicação e apresentação do modelo OSI. Estes drivers processam requisições em modo usuário, o que implica que uma aplicação só pode executar tarefas que não ofereça risco ao sistema, não podendo, por exemplo, fazer acesso direto ao hardware.

Os sistemas de arquivos permitem o acesso aos recursos do sistema, como uma chamada de leitura de uma operação de entrada e saída de dados para uma partição do tipo *NT File System* (NTFS). Cada sistema de arquivo possui um driver próprio para partições locais, como *File Allocation Table* (FAT), *High Performance File System* (HPFS) e NTFS. Maiores detalhes sobre a implementação e uso de drivers de sistemas de arquivos do Windows NT são encontrados em [Ddk96b, Bak96, Stu95].

A comunicação entre computadores em rede pode ser realizada utilizando um "redirecionador" [Ddk96a] que simula um driver de sistema de arquivos. O redirecionador se comporta como se estivesse se comunicando com um sistema de arquivos local e, no entanto, se comunica com os dispositivos remotos. A utilização deste recurso não é aplicada na implementação do protocolo proposto.

Num nível abaixo dos drivers de sistema de arquivos estão os drivers de transporte. Estes drivers implementam protocolos específicos que determinam as regras de comunicação entre computadores. Cada driver de transporte é responsável pela implementação de um protocolo específico de rede como TCP/IP ou IPX/SPX. Estes drivers são independentes dos dispositivos de hardware da camada subjacente e usam a camada limítrofe NDIS para trocar dados com o driver da adaptadora de rede e transferir pacotes sobre uma ou mais conexões físicas [Bak96].

O Windows NT permite a inclusão e utilização de drivers que não são oferecidos pelo próprio sistema operacional. Estes drivers são desenvolvidos por terceiros e chamados "não nativos". A figura 2.1 destaca o contexto em que se incluem.

Todos os drivers de transporte do Windows NT oferecem seus serviços em modo kernel para os clientes em aplicações do usuário (camadas acima), através da *Interface de Driver de Transporte* (TDI).

Os drivers de dispositivos trabalham com os aspectos específicos do hardware. Para a comunicação em redes, os dispositivos mais comumente utilizados são as adaptadoras de rede e de rádio [SLC97]. Neste texto, adaptadora de rádio é referenciada como adaptadora de rede, devido as suas funcionalidades em comum.

2.2 Camadas limítrofes

A TDI (*Transport Driver Interface*) [Ddk96a, Ddk96b] é a camada limítrofe que permite a troca de dados entre os drivers de sistemas de arquivos e os drivers de transporte, como é ilustrado na figura 2.1. Trabalha em modo kernel do sistema operacional, ou no anel 0 [MM96, Bak96], permitindo uma tarefa executar instruções privilegiadas e ter acesso completo a qualquer dispositivo de entrada e saída.

A NDIS (*Network Driver Interface Specification*) [Ddk96a, Ddk96b] é uma especificação de interface que define um conjunto de primitivas de comandos (bibliotecas de rotinas) que padronizam as regras de ligações oferecidas pelos drivers de adaptadoras de rede. Todas as interações entre um driver da adaptadora de rede com os drivers de transporte, o sistema operacional, ou adaptadoras de rede, são controladas através de chamadas de funções NDIS [Ddk96a].

A figura 2.2 ilustra a arquitetura multiprotocolar permitida pelo uso das rotinas disponibilizadas pela camada limítrofe NDIS. Cada um dos drivers de transporte, representados pelos protocolos A, B e C, pode estar ligado a mais de um driver da adaptadora de rede. Estes últimos são representados na figura pelos drivers I, J e K. A camada limítrofe NDIS trabalha como uma camada intermediária permitindo que um driver da adaptadora de rede possa ser associado indiretamente a mais de um protocolo de rede. Os drivers da adaptadora de rede são ligados a esta camada que é ligada aos drivers de transporte. Neste caso, quando um pacote é recebido por um driver J da adaptadora de rede, ele o entrega à camada limítrofe NDIS que tenta seqüencialmente entregar o pacote a cada um dos protocolos, até que um deles o aceite [SLC97]. Maiores detalhes sobre o funcionamento dos drivers e camadas limítrofes citados são encontrados em [Ddk96b, Bak96, Stu95].

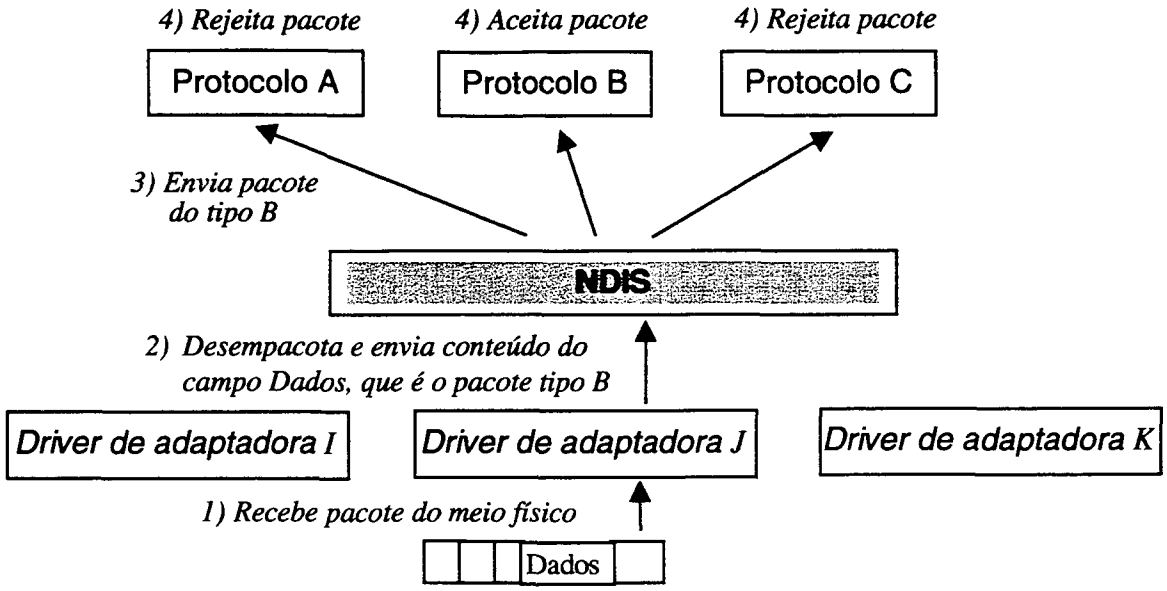


Figura 2.2 – Tratamento pelo NDIS de pacotes recebidos do meio físico de transmissão.

2.3 Detalhamento do uso de soquetes no Windows NT

Rotinas genéricas (API's) usam soquetes no ambiente de rede Windows NT, para que a aplicação do usuário seja independente do protocolo de transporte.

No uso de múltiplos protocolos de transporte, o provedor de serviços de soquetes é responsável por resolver para qual protocolo deve ir a solicitação da aplicação e recebe a chamada do soquete, direcionando para o protocolo a ser utilizado.

Funções que fazem o intercâmbio entre as rotinas genéricas dos soquetes e as rotinas específicas do protocolo devem ser fornecidas em uma biblioteca de auxílio aos soquetes, chamada *Windows Sockets Helper* (WSH), pelo desenvolvedor do driver de transporte. As bibliotecas auxiliadoras são diferentes para os diversos protocolos para resolver as ambigüidades entre eles. Com isto, uma determinada função genérica da biblioteca de soquetes pode ter várias outras correspondentes nos drivers de transporte disponíveis. Maiores detalhes para implementação da biblioteca auxiliadora de soquetes são encontrados no *Capítulo 4* e em [Ddk96a, Ddk96b, Sdk96].

A figura 2.3 apresenta o fluxo descrito e destaca a biblioteca auxiliadora de soquetes e o driver de transporte não nativos como componentes necessários para criar um protocolo de transporte independente do hardware da adaptadora de rede e poder ser utilizado em diversos meios de transmissão. Para tanto, o código de tratamento de um protocolo não nativo deve ser isolado em um módulo que trabalhe nas camadas de rede e transporte do modelo OSI.

Informações mais detalhadas sobre o funcionamento interno do Windows NT, com seus sub-sistemas gerenciadores de tarefas, são apresentadas no *Apêndice A*.

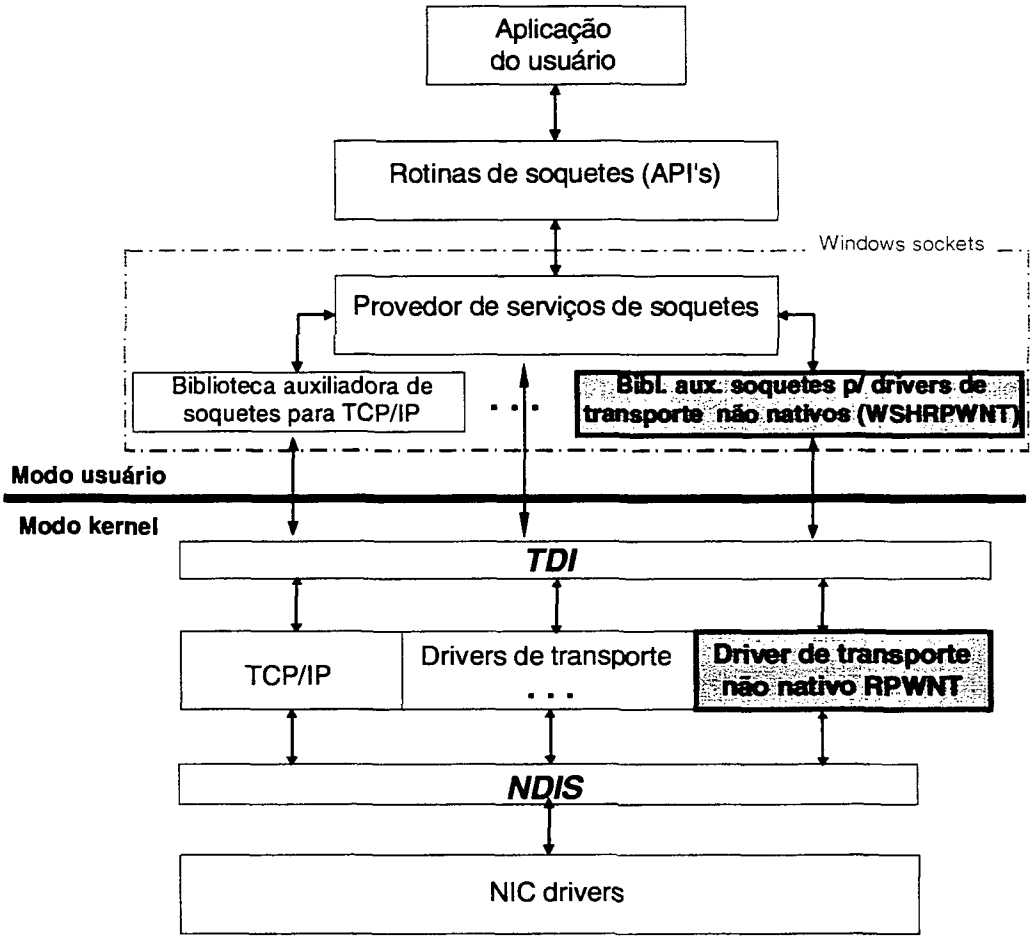


Figura 2.3 – Uso de soquetes no Windows NT.

2.4 Considerações finais

Este capítulo apresentou a arquitetura de rede do Windows NT. Destaca-se a importância dos drivers de transporte, da biblioteca auxiliadora de soquetes e das camadas limítrofes TDI e NDIS, visualizadas nas figuras 2.1 e 2.3.

Para a adaptação no ambiente de rede Windows NT e para provocar a independência do hardware da adaptadora de rede, o protocolo RPWNT proposto foi definido como um protocolo de transporte. As figuras 2.1 e 2.3 destacam o protocolo de transporte não nativo RPWNT, indicando sua localização na arquitetura de rede do Windows NT.

Para a implementação de protocolos de transporte, dois módulos devem ser desenvolvidos. Estes módulos são o driver de transporte e a biblioteca auxiliadora de soquetes, como indicados em destaque na figura 2.3. As camadas limítrofes são importantes por serem a interface de comunicação entre o protocolo de transporte e o restante do sistema operacional, ou seja, o protocolo de transporte deve utilizar funções e estruturas de dados das camadas limítrofes NDIS e TDI para ser adaptado ao Windows NT

CAPÍTULO 3

PROTOCOLOS E TECNOLOGIAS DE REDE SEM FIO

Este capítulo apresenta alguns protocolos e tecnologias para uso em comunicação de redes sem fio. Os trabalhos relacionados que contribuíram com este estudo são citados. Vantagens e desvantagens de cada protocolo e tecnologia são incluídas neste capítulo.

Para que a especificação do protocolo proposto pudesse ser definida, um estudo de diversos protocolos e tecnologias para redes sem fio foi realizado, como o objetivo de observar características relevantes para serem utilizadas em rádio pacote.

Segundo Siqueira, em [Siq96], um protocolo é um conjunto de regras de comunicação e de padrões de conexão pelas quais dois computadores trocam informações. Em rádio pacote, a troca de informações é realizada pelo compartilhamento da largura de banda do canal de rádio freqüência entre estações de rádio amador. Cada estação envia uma rajada de pacotes somente quando existe a necessidade de transmitir dados [Rob78].

As características específicas examinadas nos protocolos e tecnologias, consideradas importantes na comunicação de rádio pacote, foram: o baixo custo de implementação e utilização, a independência do hardware utilizado, o método de acesso ao meio utilizado, o tamanho do pacote para não congestionar o meio físico de transmissão, a garantia de entrega correta de pacotes através do controle de fluxo, o tratamento de erros, o tratamento de colisões e a utilização de roteamento de pacotes.

Os protocolos estudados foram: Ethernet 802.11, PPP, CDPD, TCP/IP e AX.25, e as tecnologias foram: FDMA, TDMA e CDMA. A escolha dos padrões citados ocorreu através das indicações nas referências bibliográficas de protocolos e tecnologias consideradas comuns na comunicação em rede sem fio. As seções seguintes apresentam os protocolos e tecnologias estudadas.

3.1 Protocolo Ethernet 802.11

O protocolo ETHERNET 802.11 [I3e97, Che94, CL99] é a especificação padrão para comunicação em redes locais sem fio da IEEE. Trabalha nas camadas física e de enlace do modelo OSI, sendo dependente do hardware utilizado. O pacote utilizado nas transmissões é idêntico ao do protocolo Ethernet 802.3, especificado para ser empregado em rede locais com fio [MSA86, Che94].

Estes dois protocolos utilizam o método de acesso ao meio *Carrier Sense with Multiple Access* (CSMA) [Tan94, MSA86] para permitir que estações compartilhem o mesmo meio de transmissão. Uma estação verifica se o canal está ocupado por outra estação, para iniciar o envio de dados. Se o canal está ocupado, a estação espera até que esteja disponível. Duas estações podem verificar a utilização do canal ao mesmo tempo e perceber que este está disponível, sendo possível que transmitam ao mesmo tempo. Quando isto ocorre, acontece *colisões* entre pacotes enviados pelas diferentes estações.

O protocolo Ethernet 802.3 utiliza a técnica de detecção de colisão (CD – *Collision Detection*) [Tan94, MSA86] enviando um sinal de aviso através da rede para notificar todas as estações sobre a colisão ocorrida, para que estas estações esperem por um período de tempo aleatório antes de retransmitir seus pacotes. Em ambientes de ondas de rádio, não consegue detectar a ocorrência de colisões.

O protocolo Ethernet 802.11 utiliza o mecanismo de colisão evitável (CA – *Collision Avoidance*) para reduzir a probabilidade de colisões. Após um pacote ser transmitido, é iniciado um tempo entre pacotes onde ninguém pode transmitir. Isto é, após este tempo a estação que necessita enviar pacotes inicia uma contagem de tempo aleatório que vai sendo decrementado. Com este tempo chegando a zero, a estação tenta transmitir, verificando novamente o meio. Se não estiver ocupado, transmite o pacote, senão a contagem de tempo aleatório é interrompida, devendo aguardar o final da transmissão e do tempo entre pacotes, para então obter um novo contador de tempo aleatório. A grande dificuldade deste protocolo, quando utilizado em ambientes de rádio pacote, é a negociação inicial e utilização do tempo entre pacotes, dificultada pelas freqüências de rádio lentas e com retardos [GPG98]. Não permite roteamento de pacotes, mas faz retransmissões de pacotes defeituosos e trabalham

com 3 níveis de prioridades de transmissão [DN95]. Em se tratando de controle de fluxo, este protocolo não garante a entrega de pacotes de forma ordenada.

3.2 Protocolo PPP

O *Point-to-Point Protocol* [Tan94] é um protocolo de enlace de dados que utiliza comunicações ponto a ponto entre dois computadores. Sua especificação visou a conexão de computadores com a Internet, mas aceita diferentes protocolos acima dele. É um melhoramento do protocolo SLIP [Tan94], específico para TCP/IP. O protocolo PPP detecta erros, permite que endereços IP sejam negociados em tempo de conexão e pode realizar autenticação.

O formato do pacote utilizado pelo protocolo PPP é o mesmo do protocolo HDLC, sendo que o campo Controle tem valor fixo, indicando que não faz controle de fluxo nem garante a entrega correta de pacotes. Em ambientes ruidosos, como redes sem fio, pode ser utilizada a transmissão confiável que garante a entrega de pacotes. O tamanho máximo do campo de dados é negociável na conexão, caso contrário, o tamanho utilizado é de 1500 bytes .

3.3 Tecnologia FDMA

A tecnologia FDMA (*Frequence Division Multiple Access*) [DN95, RBP95] é um padrão para uso em telefonia celular analógica e é baseado em multiplexação de freqüência de canais de rádio freqüência (FDM). Este protocolo opera em modo full-duplex, transmitindo e recebendo dados em duas freqüências diferentes e únicas, negociadas entre cada usuário da célula e a estação central, o que resulta na inexistência de colisões de pacotes, mas podendo ocorrer interferências na faixa que está sendo utilizada por um usuário (ruído), danificando pacotes.

Uma célula é uma área geograficamente limitada pelo alcance da transmissão realizada pela estação central. Cada célula da rede possui uma torre de transmissão que liga o usuário móvel ao serviço de telefonia fixa, e a transmissão é dependente das regiões que possuem

cobertura das células. A implantação de uma torre de transmissão, para criar uma nova célula, é um processo caro e só pode ser realizado pela empresa concessionária autorizada.

Este sistema é ligado à telefonia fixa para enviar dados a longa distância [Che94]. Um exemplo de utilização desta especificação CDPD é o AMPS (*Advanced Mobile Phone Service*), utilizado no Brasil.

Este protocolo realiza o tratamento de erros referente às reflexões, que fazem o sinal chegar ao receptor com atrasos, através de comparações e compensações de múltiplos sinais recebidos. Os modens utilizados para a transmissão de dados e que transformam sinais analógicos em digitais, realizam tratamentos de erros complementares. O protocolo permite o roteamento de pacotes, possibilitando que a comunicação não se interrompa no deslocamento do usuário de uma célula a outra.

As comunicações que utilizam telefonia, tanto móvel quanto fixa, criam conexões ponto-a-ponto entre estações, não permitindo o envio de dados por *broadcast* e comunicações multiponto, como possibilitados em rádio pacote [Siq96].

3.4 Tecnologia TDMA

A tecnologia TDMA (*Time-Division Multiple Access*) [RBP95, DN95, Cap79, Che94]. é um padrão digital para telefonia celular baseada na técnica em que um mesmo canal é usado por três usuários, um de cada vez, através de multiplexação de tempo (TDM). Cada usuário tem a seu dispor toda a largura de banda do canal, durante certo tempo. Cada canal do TDMA tem a mesma largura de banda dos canais do AMPS. Como no AMPS, a comunicação de dados depende das regiões que possuem cobertura das células e a expansão destas células é cara, como no FDMA e CDMA [Siq96].

Esta tecnologia permite a detecção e correção de erros através de poderosos algoritmos. Os erros referentes às reflexões são resolvidos através do espelhamento dos bits no tempo segundo uma seqüência conhecida. A tecnologia realiza o controle de fluxo de dados e permite o roteamento de pacotes, possibilitando que a comunicação não se interrompa no

deslocamento do usuário de uma célula a outra. Cria conexões ponto-a-ponto entre estações, não permitindo o envio de dados por *broadcast* e comunicações multiponto [Siq96].

3.5 Tecnologia CDMA

A tecnologia CDMA (*Code-Division Multiple Access*) [RBP95, Siq96, Che94] é um padrão digital para telefonia celular baseado em difusão do espectro (*Spead Spectrum*), onde um transmissor espalha ou difunde o sinal de rádio sobre uma ampla gama de freqüências, segundo uma seqüência determinada. No lado da recepção o sinal só pode ser detectado por receptores de faixa larga e que conheçam a seqüência de espalhamento. Todos os telefones móveis transmitem seus sinais ao mesmo tempo e possuem um código binário exclusivo para diferenciar um do outro no lado do receptor. É muito difícil interferir ou ouvir uma transmissão via rádio que use CDMA.

Esta tecnologia multiplica a capacidade do AMPS em até 18 vezes, pois comprime o sinal digitalizado como no TDMA e utiliza uma banda de freqüência mais larga. Também neste sistema, a comunicação de dados depende das regiões que possuem cobertura das células e a expansão destas células é cara, como no FDMA e TDMA [Siq96].

O CDMA permite detecção e correção de erros no nível físico através de um código convolucinal, CRC e entrelaçamento de blocos, o que aumenta a taxa efetiva de transferência de dados do enlace. Realiza o controle de fluxo de dados e permite o roteamento de pacotes, possibilitando que a comunicação não se interrompa no deslocamento do usuário de uma célula a outra. Cria conexões ponto-a-ponto entre estações, não permitindo o envio de dados por *broadcast* e comunicações multiponto [Siq96].

3.6 Protocolos TCP e IP

Os protocolos TCP e IP [GB93, BDM95] são os padrões utilizados na rede mundial Internet. Trabalham nas camadas de rede (IP) e transporte (TCP), não dependendo do hardware utilizado e sendo considerado de baixo custo de utilização. O TCP possibilita o roteamento de

pacotes e realiza tratamentos de erros e controles de fluxo dos pacotes. É orientado à conexão, garantindo a entrega correta de pacotes. Em computação móvel, a manutenção das tabelas dinâmicas de roteamento deste protocolo é de difícil realização, pois o usuário em movimento deve alterar seu endereço IP para cada rede que se deslocar [GB93].

3.7 Protocolo AX.25

O protocolo AX.25 [BN97, Tar97, Wat97] (do inglês *Amateur X.25*) teve sua primeira versão desenvolvida em outubro de 1982 e foi baseado no protocolo X.25 [Ren82]. O protocolo X.25, por sua vez é baseado no protocolo HDLC [Tan94, MSA86] definido pela ISO, e que trabalha nas camadas física e de enlace. O protocolo X.25 foi alterado para permitir as adequações de endereçamento e transmissão via rádio [Tar97], aproveitando o conceito de circuito virtual entre estações comunicantes, ou seja, uma conexão direta entre dois pontos é realizada, dando a impressão de um circuito único entre eles.

O protocolo AX.25 proporcionou a interligação de milhares de estações repetidoras de rádio amador formando uma rede de propagação global que consiste em diversos tipos de nós, como gateways de alta-freqüência (HF), gateways de Internet e ligações via satélite [BN97]. Os gateways são usados para permitir a transmissão de dados entre freqüências diferentes, sendo pontos de ligação entre a rede de pacotes e a Internet.

Este protocolo foi elaborado para o ambiente de rádio pacote e utiliza freqüências de rádio lentas e com retardos, mas de baixo custo. Trabalha nas camadas físicas e enlace, entretanto possui funções das camadas de rede e transporte. Usa o mecanismo de contenção CSMA. É um protocolo orientado à conexão, garantindo a entrega correta dos pacotes através do reconhecimento de recebimento dos mesmos e utiliza janelas deslizantes para controle de fluxo e tratamento de erros [BN97].

Uma grande vantagem é a capacidade de utilização de estações de rádio como repetidoras de pacotes, aumentando a distância da comunicação, uma vez que os computadores origem e destino não necessitam estar próximos. O tamanho do campo de dados do pacote é de 256 bytes. Permite o roteamento estático, indicando um número indeterminado de endereços de estações repetidoras no pacote. Utiliza comunicações half-duplex e full-duplex. Permite o

compartilhamento do canal e realiza conexões do tipo multiponto, comunicando-se com várias estações ao mesmo tempo.

Por apresentar características que melhor se adaptam ao ambiente de rádio pacote, o protocolo AX.25 foi utilizado como base para a especificação do protocolo RPWNT proposto a ser implementado no ambiente de rede Windows NT.

3.8 Considerações finais

Este capítulo apresentou o estudo de alguns protocolos e tecnologias para comunicação em redes sem fio. Vantagens e desvantagens em relação ao uso em ambiente de rádio pacote foram mencionadas e são apresentadas no quadro comparativo 3.1.

PROTOCOLO	Baixo custo	Independência do hardware	Método de acesso	Tamanho do campo de dados	Garantia de entrega	Tratamento de erros	Roteamento	Específico p/ rádio pacote
Ethernet 802.3	Sim	Não	CSMA/CD	40-1500	Não	Não	Não	Não
Ethernet 802.11	Sim	Não	CSMA/CA	40-1500	Não	Parcial	Não	Não
PPP	Sim	Não	Próprio	1500	Não (opcional)	Sim	Não	Não
FDMA (CDPD)	Não	Não	Baseada em FDM	Não se aplica	Não	Parcial (modens)	Sim	Não
TDMA	Não	Não	Baseada em TDM	Não se aplica	Sim	Sim	Sim	Não
CDMA	Não	Não	Baseada em Spread Spectrum[1]	Não se aplica	Sim	Sim	Sim	Não
TCP/IP	Sim	Sim	Não se aplica	1460	Sim	Sim	Estático[2], dinâmico[2]	Não
AX.25	Sim	Não	CSMA	0-256	Sim	Sim	Estático	Sim

Quadro 3.1 – Comparação entre protocolos e tecnologias que utilizam rede sem fio.

[1] Um transmissor espalha ou difunde o sinal de rádio sobre uma ampla gama de freqüências, segundo uma seqüência determinada. No lado da recepção o sinal só pode ser detectado por receptores de faixa larga e que conheçam a seqüência de espalhamento [RBP95].
[2] Definições apresentadas por Tanenbaum [Tan94].

Os padrões estudados foram escolhidos através de indicações nos trabalhos relacionados a seguir, e as características observadas são relevantes na utilização em rádio pacote.

Chen, em [Che94], realiza uma abordagem de serviços de computação móvel para redes locais que utilizam multicélulas, e que necessitam de um controle efetivo de acesso ao meio para garantir a velocidade destes serviços. Apresenta o método de acesso CSMA e o protocolo Ethernet 802.11 que são utilizados descritos neste trabalho.

Rochol et al., em [RBP95], fazem uma abordagem comparativa em relação às principais técnicas utilizadas em telefonia celular FDMA, TDMA e CDMA para Redes Celulares de Telefonia Móvel (RCTM). São analisados e justificados os principais critérios de comparação destas tecnologias, tais como: capacidade máxima por célula, imunidade a ruído, robustez quanto a caminhos múltiplos de recepção, interferência e "fading", segurança e sigilo, capacidade de suportar os atuais serviços de comunicação de dados e os novos serviços de computação móvel. Algumas características estudadas e resultados são utilizados neste trabalho, como o método de acesso utilizado, o tratamento de erros e colisões e utilização de roteamento.

Tanenbaum, em [Tan94], apresenta o protocolo PPP muito utilizado para acesso à Internet utilizando modems e linhas telefônicas com acesso por discagem. Destaca as características deste protocolo e faz um comparativo com o protocolo SLIP, ressaltando os pontos positivos do protocolo PPP. Pode ser utilizado em conjunto com tecnologias como FDMA, TDMA e CDMA para comunicações sem fio. Alguns conceitos foram utilizados neste trabalho.

Lam et al, em [LL81], apresentam um estudo experimental que foi conduzido utilizando um simulador de rede que investiga a performance da comunicação que usam pacotes. Algumas funções verificadas por este simulador são: capacidade da rede, controle de fluxo, janelas deslizantes e roteamentos realizados por protocolos. Alguns resultados da investigação e conceitos foram aproveitados.

Gasparini et al, em [GB93], apresentam a arquitetura e funcionamento do protocolo TCP/IP e os serviços de conectividade e atribuição de nomes. Cita algumas vantagens e desvantagens no uso em computadores móveis. Estas informações foram utilizadas no estudo sobre protocolos para rede sem fio realizado.

Uma proposta de implementação e otimização da utilização do protocolo TPC/IP sobre o protocolo AX.25 é descrita e discutida por Gunter em [Gun97]. Num primeiro momento, o protocolo TCP/IP apresenta baixa performance e ineficiência no uso em rádio pacote, por não ser específico para este ambiente. Algumas dificuldades do uso do protocolo TCP/IP no ambiente de rádio pacote são citadas e otimizações são sugeridas.

O documento [BN97] é uma revisão do protocolo AX.25 versão 2.0 disponibilizado pelo Comitê de Comunicações Digitais da ARRL (*American Radio Relay League*). Contém a especificação e as funcionalidade do protocolo AX.25, que foi utilizado como base da especificação do protocolo de rádio pacote proposto.

CAPÍTULO 4

O PROTOCOLO DE TRANSPORTE RPWNT

Este capítulo propõe a especificação de um protocolo para comunicação de rádio pacote a ser utilizado no ambiente de rede Windows NT. O protocolo proposto neste trabalho é denominado RPWNT – Rádio Pacote para Windows NT, baseado no protocolo AX.25 e é descrito a seguir. As principais características deste protocolo são apresentadas na seção 4.1, seguindo-se da proposta dos tipos de pacotes utilizados pelo RPWNT e suas finalidades, na seção 4.2. Na seção 4.3 são determinadas as estruturas destes pacotes e seus campos. Os pacotes de comandos e pacotes de respostas são apresentados na seção 4.4, juntamente com os parâmetros deste protocolo. Os procedimentos de tratamento de erros e o controle da numeração dos pacotes são descritos nas seções 4.5 e 4.6. Por fim, na seção 4.7 são apresentados os procedimentos envolvidos na comunicação entre duas estações de rádio e um exemplo de comunicação entre estas estações.

4.1 Características do protocolo RPWNT

As principais características do protocolo RPWNT são apresentadas a seguir. É importante observar que estas características são comuns ao protocolo AX.25.

- Os endereços das estações de rádio origem e destino utilizam o código de licença de uso do usuário, que é único no mundo. Este código é conhecido como *callsign*;
- Permite um alcance maior na comunicação através do enfileiramento das estações de rádio, que podem ser usadas como repetidoras, também conhecidas como *digipeaters*;
- Cada pacote pode conter um número indeterminado de endereços de repetidoras, indicando por quais já passou e por quais ainda passará;

- Permite que várias estações de rádio sejam conectadas e se comuniquem simultaneamente;
- É um protocolo orientado à conexão, que garante a entrega correta dos pacotes através do aviso reconhecimento de recebimento dos mesmos. Este aviso é conhecido como *ack* (*acknowlegment*);
- Faz controle de fluxo através de pacotes de supervisão como: pronto para receber, temporariamente ocupado, recepção com sucesso (ack), retransmissão do n-ésimo pacote em diante, retransmissão do pacote específico, teste de conexão, estabilização e término de conexão;
- Utiliza janelas deslizantes, que é um esquema de recepção de grupos de pacotes, com uma única confirmação após o último pacote deste grupo ter chegado. Utiliza uma lista da seqüência dos pacotes que devem ser recebidos. Diminui o tráfego de dados, pois não tem que confirmar a chegada com sucesso de cada pacote. Com isto, quando pacotes são perdidos, permite a retransmissão de um único, ou de um pacote indicado em diante, dentro do intervalo da janela deslizante;
- Permite o envio de aviso de ack's embutidos em pacotes de dados, diminuindo o tráfego;
- Utiliza o cálculo de seqüência de verificação dos pacotes, o FCS – *Frame Check Sequence*, que garante a recepção de pacotes que não estejam corrompidos;
- Realiza tratamentos de erros, como detecção e recuperação, fornecendo uma comunicação livre de erros. Alguns destes tratamentos ocorrem quando a estação destino está ocupada, o número de seqüência de envio do pacote está incorreto, o pacote é rejeitado, o limite de tempo de espera para receber um ack está esgotado (*time-out*), o FCS é inválido e os acks são perdidos;
- Ajusta automaticamente o tempo de time-out no momento da conexão;
- Faz manutenção do enlace com baixa ou nenhuma taxa de transferência, para manter a conexão ativa.

Para o usuário final as operações na comunicação são transparentes, como conexão, escrita da mensagem e envio automático. A estação de rádio, conhecida como *Terminal Node Controller* (TNC), divide a mensagem em diversos pacotes, chaveia o transmissor correto e

envia estes pacotes. Quando o TNC recebe pacotes, ele automaticamente os decodifica, verifica os erros e monta a mensagem recebida, ordenando os pacotes.

Verifica-se que a melhor aplicação para este protocolo está nos ambientes de transmissão com maior índice de perdas de pacotes, devido a seu tratamento de erros, garantindo a entrega de pacotes. Nos ambientes com baixo índice de perdas de dados, os protocolos com menos quantidade de tratamentos de erros garantem a entrega dos pacotes com maior rapidez. No entanto, o protocolo proposto deve trabalhar em qualquer destes ambientes.

4.2 Tipos de pacotes

Durante a comunicação, o protocolo RPWNT pode utilizar três tipos de pacotes, que são de informação, supervisão e manutenção, sendo este último definido com não-numerado, como segue :

- **Informação**: contém os dados em transmissão ou a serem transmitidos que são passados pela aplicação do usuário, podendo levar um ack embutido para diminuir o tráfego da comunicação. A estrutura do pacote é ilustrada na figura 4.1 e seus campos são explicados na seção 4.3. O tamanho mínimo do pacote é de 20 bytes e o máximo varia de acordo com o número de endereços de estações repetidoras, representado na figura por "*".

- **Supervisão**: contém informações de supervisão e controle de fluxo de dados, como por exemplo os acks e solicitações de retransmissão de pacotes. Maiores detalhes são apresentados na seção 4.4. A estrutura do pacote é ilustrada na figura 4.2 e seus campos são explicados na seção 4.3. O tamanho mínimo do pacote é de 19 bytes e o máximo varia de acordo com o número de endereços de estações repetidoras, representado na figura por "*".

- **Não Numerado**: é responsável pela estabilização e término de conexões, o envio e o recebimento de informações sobre um computador, a negociação de parâmetros da comunicação e os testes do funcionamento do enlace. O controle de fluxo não é realizado para este tipo de pacote. Maiores detalhes são apresentados na seção 4.4. A estrutura do pacote é ilustrada na figura 4.2 e seus campos são explicados na seção 4.3. O tamanho

mínimo do pacote é de 19 bytes e o máximo varia de acordo com o número de endereços de estações repetidoras, representado na figura por "*".

4.3 Estruturas dos pacotes

As figuras 4.1 e 4.2 a seguir apresentam respectivamente as estruturas dos pacotes de informação e de supervisão e não numerados.

Flag	Endereço Destino	Endereço Origem	Endereços das Repetidoras	PID	Controle	Dados	FCS	Flag
1 byte	7 bytes	7 bytes	*	1 byte	1 ou 2 bytes	N bytes	2 bytes	1 byte

Figura 4.1 – Estrutura do pacote de informação do RPWNT.

Flag	Endereço Destino	Endereço Origem	Endereços das Repetidoras	Controle	Dados	FCS	Flag
1 byte	7 bytes	7 bytes	*	1 ou 2 bytes	N bytes	2 bytes	1 byte

Figura 4.2 – Estrutura dos pacotes de supervisão e não numerados RPWNT.

Os campos das estruturas de pacotes do protocolo RPWNT ilustradas nas figuras 4.1 e 4.2 são apresentados a seguir.

a) **Flag** : com tamanho de 1 byte, delimita o pacote, indicando seu início e término. O valor fixo é "01111110".

Para garantir que a seqüência de bits do campo FLAG não apareça acidentalmente em qualquer parte do pacote, indicando o início ou fim do pacote através da seqüência do envio de um grupo de 5 ou mais bits "1" contínuos, o protocolo implementa o recurso *bit stuffing*. Na criação do pacote, onde aparece 5 ou mais bits "1" seqüenciais, é inserido um bit "0" depois do quinto bit "1". Durante o recebimento do pacote, para qualquer seqüência de 5 bits "1", o bit "0" imediatamente seguinte é descartado [BN97].

b) **Endereço destino** : com tamanho de 7 bytes, possui 6 bytes para o endereço da estação de rádio, o callsign, e 1 byte para identificação secundária (SSID) descrita a seguir. Se o callsign for menor que o determinado, o subcampo é preenchido com espaços. Este endereço pode ser um nome de grupo, criando uma operação "ponto-para-multiponto". O caminho que o pacote deve percorrer para alcançar todas as estações de um grupo é o mesmo, sendo que estas estações devem estar próximas;

Os 8 bits do subcampo de identificação secundária (SSID) são representados por *"C R R S S I D O"* :

- **Bit "C"** [1] : os bits "C" do destino e origem, em conjunto, indicam a versão do protocolo, sendo Velha = "0" ou Nova="1". Para os endereços das repetidoras, este bit indica se o pacote já passou (valor 1) ou não (valor 0) pela repetidora;;
- **Bits "R R"** : bits reservados. Poder ser utilizados para distinção entre redes individuais;
- **Bits "S S I D"** : identificador de estação secundária. Algumas aplicações requerem mais de um endereço em uso e o callsign pode ser utilizado em conjunto com este campo.
- **Bit "O"** : indica se existe próximo endereço para repetidora, criando uma lista encadeada de endereços de repetidoras, sendo Sim = "0" e Não = "1";

c) **Endereço origem** : com tamanho de 7 bytes, possui 6 bytes para o endereço da estação de rádio, o callsign, e 1 byte para identificação secundária (SSID) descrita anteriormente. Se o callsign for menor que o determinado, o subcampo é preenchido com espaços;

d) **Endereços das repetidoras** : cada endereço de repetidora possui tamanho de 7 bytes, sendo 6 bytes para endereço da estação de rádio, o callsign, e 1 byte para identificação secundária (SSID) descrita anteriormente. Se o callsign for menor que o determinado, o subcampo é preenchido com espaços. O tamanho mínimo para este campo é 0 bytes, caso

---

[1] O item indicado é especificado pelo protocolo RPWNT para manter a compatibilidade com o protocolo AX.25, mas não possui utilização no protocolo proposto. Mais detalhes sobre a utilização deste campo pelo protocolo AX.25 são encontrados em [BN97];

não haja repetidoras. O sinal "*" nas figuras 4.1 e 4.2 indica que este campo pode conter uma lista de endereços de estações repetidoras, sem limite máximo;

e) **PID** [1] : indica qual o tipo de protocolo da camada de rede está sendo usado;

f) **Controle** : identifica os três tipos de pacotes suportados pelo protocolo: informação, supervisão e não numerado. A escolha pelo tamanho deste campo, 1 byte ou 2 bytes, no momento da conexão entre as estações indica o tamanho que será utilizado pela janela deslizante. Se o tamanho escolhido do campo de controle é de 1 byte, o tamanho da janela deslizante será de 8 pacotes. Se o tamanho escolhido for de 2 bytes, o tamanho da janela deslizante será de 128 pacotes. A figura 4.3 ilustra o campo "Controle" e seus subcampos, em seu formato padrão de tamanho 1 byte, separados por tipo de pacote.

Tipo do pacote	**B**			**I**	**T**	**S**		
	7	6	5	4	3	2	1	0
Pacote informação	N (R)			P	N (S)			0
Pacote supervisão	N (R)			P/F	S	S	0	1
Pacote não numerado	M	M	M	P/F	M	M	1	1

Indicam o tipo do pacote

Figura 4.3 – Tipos de pacotes do protocolo RPWNT, definidos no campo CONTROLE.

No pacote informação, o subcampo N(S) indica número do pacote que está sendo enviado. O subcampo N(R) indica o número do pacote esperado no retorno, pelo computador que está enviando este pacote. Estes números são utilizados para os controles de seqüência da janela deslizante. O subcampo P, se igual a "1", indica que um ack está sendo levado neste pacote.

No pacote supervisão, os bits "SS" indicam os comandos e respostas enviados por este pacote e descritos no item 4.3. O bit P ou F, "1" ou "0" respectivamente, indica se o pacote é um "comando" de solicitação de tarefa ou uma "resposta" para esta solicitação. O subcampo N(R) indica o número do pacote esperado no retorno, pelo computador que está enviando este pacote.

No pacote não numerado, os bits "MMM MM" indicam os comandos e respostas que podem ser enviados por este pacote. O bit P ou F, "1" ou "0" respectivamente, indica se o pacote é um comando de solicitação de tarefa ou uma resposta para esta solicitação

g) **Dados** : dados provenientes da aplicação do usuário a serem transmitidos. O tamanho padrão é de 256 bytes, podendo ser renegociado no momento da conexão, até este valor máximo de 256 bytes. Usado em pacotes do tipo informação e em alguns casos no tipo não numerado, apresentados na seção 4.4;

h) **FCS** – *Frame Check Sequence* (Seqüência de Verificação do Pacote): contém o valor resultante do cálculo realizado sobre todos os bytes do pacote para garantir que este pacote não seja corrompido pelo meio de transmissão. O cálculo é feito tanto no transmissor quanto no receptor e utiliza a norma ISO3309 [BN97].

4.4 Pacotes de comandos e pacotes de respostas

Para a supervisão do controle do fluxo de dados, o tratamento de erros e manutenção do enlace, o protocolo RPWNT utiliza pacotes que enviam solicitações de tarefas, os comandos, e as respostas para estas solicitações. Os valores dos bits são relacionados ao item "f" da seção 4.3. Dependendo da situação, os seguintes itens disponibilizados para os pacotes do tipo supervisão podem ser comandos ou respostas.

- Pronto para receber (**RR** – *Receive Ready*) = "00" : indica que a estação de rádio está pronta para receber pacotes. Em conjunto com o número do próximo pacote a ser recebido N(R), indica um ack;
- Não pronto para receber (**RNR** – *Receive Not Ready*) = "01" : estação de rádio temporariamente ocupada. Os pacotes enviados após este pacote serão descartados e deverão ser retransmitidos;
- Rejeição (**REJ** – *Reject*) = "10" : retransmissão do pacote indicado pelo N(R) e posteriores;
- Rejeição seletiva (**SREJ** – *Selective Reject*) = "11" : retransmissão somente do pacote indicado por N(R);

Para os pacotes do tipo não numerado, os seguintes comandos e respostas estão disponíveis :

- (**SABM** – *Set Async Balanced Mode*) = "001 11" : comando para requisição de conexão, utilizando o padrão de 1 byte para o campo de controle;
- (**SABME** – *Set Async Balanced Mode Extended*) = "011 11" : comando para requisição de conexão utilizando 2 bytes para o tamanho do campo de controle;
- Desconexão (**DISC** – *Disconnect*) = "010 00" : comando para requisição de desconexão, para encerrar o enlace;
- Reconhecimento não numerado (**UA** – *Unnumbered Acknowledge*) = "011 00" : resposta de reconhecimento de recepção de pacotes do tipo SABM, SABME e DISC;
- Modo de desconexão (**DM** – *Disconnect Mode*) = "000 11" : resposta de sistema ocupado ou desconectado (*Disconected Mode*);
- Informação não numerada (**UI** – *Unnumbered Information*) = "000 00" : pacote fora do fluxo de controle que solicita informações sobre a situação do TNC. Pode receber como resposta DM, RR ou RNR;
- Teste (**TEST**) = "111 00" : comando e resposta para testar o enlace entre estações de rádio. Utiliza o campo de dados para receber informações;
- Troca de identificação (**XID** – *Exchange Identification*) = "101 11" : pacote de troca de identificação. A estação que recebe um comando XID, envia de volta uma resposta XID com informações suas, do tipo comandos aceitos (REJ, SREJ, UI, Campo de controle de 1 ou 2 bytes, Endereços estendidos, FCS de 16/32 bits), sincronização, tamanho da janela deslizante, tamanho máximo do campo de informações, entre outras. Utiliza o campo de informações para enviar estes dados. Esta negociação de informações é realizada na conexão entre computadores e pode ocorrer em qualquer outro momento da comunicação. Os valores padrões destes parâmetros são :
  - Tempo de espera da resposta de reconhecimento de recebimento de pacotes (T1) = 3000 ms. Este parâmetro garante que a estação de rádio origem não vai esperar indefinidamente pela resposta de um pacote enviado. Sendo T o tempo gasto para um pacote percorrer a distância entre o computador origem e o destino, incluindo a passagem por estações repetidoras T1 >= 2T;

- Número máximo de tentativas = 10. Utilizado em conjunto com T1, indica o número de tentativas que devem ocorrer quando algum procedimento falha;
- Retransmissão de pacotes = Rejeição Seletiva;
- Janela Deslizante = 8 pacotes;
- Tamanho do campo de informações = 256 bytes;

O quadro a seguir apresenta as ligações entre os comandos e respostas enviados pelo protocolo RPWNT :

ESTAÇÃO DE RÁDIO ORIGEM		**ESTAÇÃO DE RÁDIO DESTINO**		
Comando Solicitado	**Descrição**	**Estado no recebimento**	**Resposta Enviada**	**Descrição**
SABM	Requisição de conexão	Transferência ou Desconectado	UA DM	Conexão aceita Sistema desconectado
DISC	Requisição de desconexão	Transferência ou Desconectado	UA DM	Desconexão aceita Sistema desconectado
S I	Todos de supervisão Pacote de dados	Desconectado	DM	Sistema desconectado
S I	Todos de supervisão Pacote de dados	Transferência	RR RNR REJ SREJ	Ack, sistema pronto Sistema temporariamente ocupado Retransmissão *n* em diante Retransmissão de *n*
XID	Troca informações do enlace	Transferência ou Desconectado	XID	Troca de informações do enlace
UI	Informação do estado da estação de rádio	Qualquer	DM RR RNR	Sistema desconectado Sistema pronto Sistema temporariamente ocupado
REJ SREJ	Retransmissão *n* em diante Retransmissão de *n*	Transferência	RR + I RNR	Sistema pronto + dados Sistema temporariamente ocupado
TEST	Solicitação de teste do enlace	Qualquer	TEST	Resposta do teste de enlace

Quadro 4.4 – Comandos e respostas do protocolo RPWNT.

4.5 Tratamentos de erros

A seguir são apresentadas algumas situações de erros que podem ocorrer durante o uso do protocolo RPWNT e os procedimentos executados por este protocolo para garantir a continuidade da comunicação:

- **Estação de rádio ocupada**: envia pacote indicando que está temporariamente ocupada (RNR) e quando estiver novamente liberada, envia um dos pacotes indicando resposta de "pronto para receber pacotes" (RR), retransmissão de pacotes (REJ), requisição de conexão (SABM / SABME) ou reconhecimento de recepção (UA);
- **Número de seqüência de envio incorreto**: guarda o pacote se o número do pacote pertencer à janela deslizante, senão descarta-o;
- **Um pacote rejeitado**: quando vários pacotes pertencentes à janela deslizante foram recebidos sem problemas, é solicitada a retransmissão somente do pacote que não foi recebido com sucesso (SREJ);
- **Dois ou mais pacotes rejeitados**: é solicitada a retransmissão do pacote indicado e posteriores (REJ);
- **Time-out**: se a estação de rádio origem não receber o ack dos pacotes transmitidos por um tempo determinado, estes são retransmitidos. Para não ocorrer o erro de time-out quando a taxa de transmissão no meio físico é baixa ou não há o que transmitir, a estação de rádio garante o funcionamento do enlace enviando pacotes esporadicamente;
- **FCS inválido**: o pacote é descartado.
- **Ack perdido**: se um ack enviado pela estação de rádio destino não chegar à estação de rádio origem, este último começa a retransmitir os pacotes que aguardam confirmação e que pertencem à janela deslizante corrente, pois ocorreu time-out. A estação de rádio destino, recebendo um pacote duplicado que já estava na janela deslizante, envia novamente um ack para a estação de rádio origem, informando que todos os pacotes desta janela deslizante já foram recebidos.

4.6 Controle da numeração dos pacotes

A figura 4.5 permite visualizar como é realizado o controle da numeração de pacotes transmitidos entre as estações de rádio. Para operações de janela deslizante com 8 pacotes, pacotes de informação recebem numeração de "0" a "7" e para operações de janela deslizante com 128 pacotes, pacotes deste tipo recebem numeração de "0" a "127".

ESTAÇÃO DE RÁDIO 1	
V(S):	*Próximo pacote a ser enviado*
V(R):	*Próximo pacote a ser recebido*

PACOTE ENVIADO	
N(S):	*N° do pacote atual*
N(R):	*Pacote esperado na volta*

ESTAÇÃO DE RÁDIO 2
V(S)
V(R)

Figura 4.5 – Controle de fluxo do protocolo RPWNT.

Nas estações de rádio, a variável V(S) indica o número do próximo pacote a ser enviado e a variável V(R) indica o número do próximo pacote a ser recebido. No pacote enviado, a variável N(S) indica o número deste pacote, e a variável N(R) indica o número do próximo pacote esperado pela estação de rádio origem.

Os procedimentos a seguir demonstram os passos realizados no momento da transmissão do pacote pela estação de rádio 1:

1) **N(R) = V(R)**. Indica que os pacotes "N(R)–1" e anteriores foram recebidos com sucesso;
2) **N(S) = V(S)**;
3) **V(S) = V(S) + 1**. Sendo LI o limite inferior da janela deslizante, e LS o limite superior da mesma janela, se V(S) > LS, a estação de rádio 1 aguarda o ack a ser recebido, tornando V(S) = LI logo após esta resposta ser recebida;

O protocolo proposto realiza os seguintes procedimentos no momento da recepção do pacote pela estação de rádio 2:

- Se **N(S) = V(R)**, então **V(R) = V(R) + 1.** Se V(R) > LS, é enviado um ack para a estação de rádio 1 e V(R) = LI.
- Se **N(S) < V(R)**, então descarta o pacote, senão solicita retransmissão deste pacote.

4.7 Descrição dos procedimentos

A figura 4.6 ilustra uma simulação da comunicação sem erros entre duas estações de rádio (ER) e os procedimentos envolvidos no estabelecimento, uso e finalização da conexão, sendo utilizado o protocolo RPWNT.

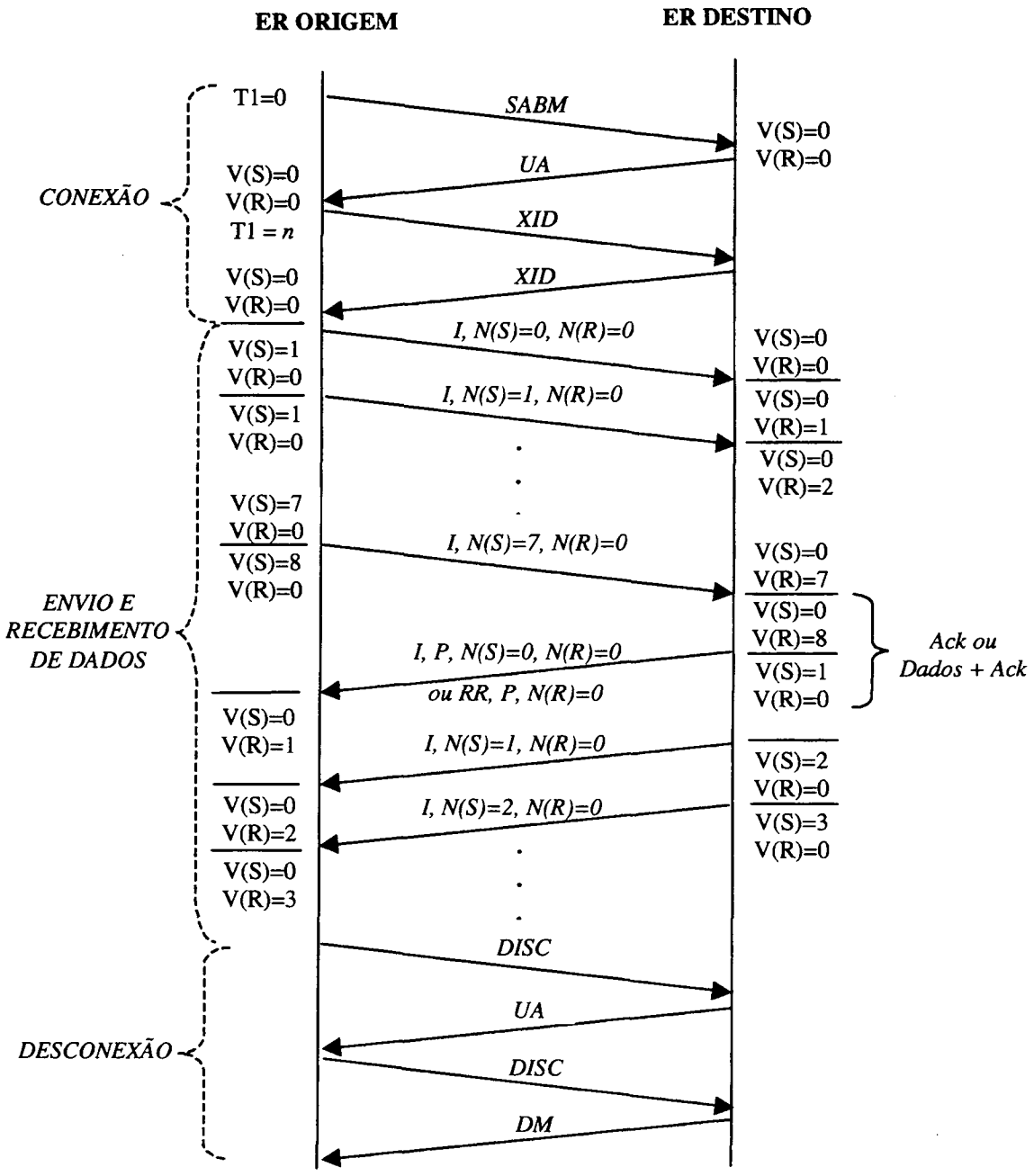


Figura 4.6 – Exemplo de comunicação usando o protocolo RPWNT.

O primeiro passo na comunicação entre computadores usando o protocolo proposto é a criação de uma conexão que irá formar um canal virtual, como é ilustrado na parte indicada como "conexão" da figura 4.6.

A ER origem recebe a solicitação de conexão da aplicação do usuário, cria e inicializa um pacote do tipo não numerado com informações sobre os endereços destino, origem e repetidoras e o comando de solicitação da conexão (SABM ou SABME). Envia este pacote para a ER destino e aguarda a resposta de conexão aceita ou destino ocupado.

A ER destino recebe o pacote do meio físico e entra no estado de "transferência de dados". Retorna para a ER origem um pacote do tipo não numerado, contendo a resposta de conexão aceita (UA) e os endereços que indicam o caminho inverso a percorrer.

Caso a ER destino não consiga entrar no estado de "transferência de dados", devido a, por exemplo, algum problema como a sobrecarga nas filas de recebimento de pacotes, então um pacote indicando que a ER está ocupada (DM) é enviado para a ER origem.

A ER, tanto origem, destino ou repetidora, verifica a integridade de todos os pacotes que são recebidos através do cálculo do FCS, sendo descartados se forem inválidos. O endereço da ER atual é comparado com o endereço destino do pacote para confirmar se este pacote chegou ao seu destino final. Se não for igual, então é verificado se o endereço da ER atual faz parte da lista de repetidoras que este pacote deve passar, sendo descartado caso contrário. Se a ER atual for a próxima estação repetidora indicada na lista pelo pacote, então ela retransmite este pacote.

Com a resposta de conexão aceita, a ER origem troca pacotes com a ER destino para a negociação dos valores padrões dos parâmetros que serão utilizados nesta comunicação. Para tanto, utilizam pacotes do tipo Troca de Identificação (XID). Ao final desta tarefa, as duas ER se encontram no estado de "transferência de dados" e estão aptas a iniciar o envio e recebimento dos pacotes de dados, como ilustrado na parte que indica "envio e recebimento de dados" da figura 4.6.

Para enviar dados recebidos da aplicação do usuário, a ER origem obtém o número do próximo pacote a ser enviado V(S), cria e inicializa o pacote do tipo informação. Se a quantidade de dados provenientes da aplicação do usuário é maior que a capacidade do

pacote, devem ser criados vários outros pacotes para o envio destes dados, seguindo a seqüência numerada V(S). Os pacotes são enviados até que o limite superior da janela deslizante seja alcançado, que neste exemplo é de 8 pacotes, devendo a ER origem aguardar um ack destes pacotes.

A ER destino recebe os pacotes enviados até que a janela deslizante seja completada. Um ack é enviado à ER origem através de um pacote do tipo supervisão com um comando de pronto para receber (RR), ou embutido em um pacote de dados.

Para cada pacote que recebe, a ER destino verifica se o número do pacote recebido N(S) está contido nos limites da janela deslizante, descartando-o se não estiver. Compara o N(S) com o número do pacote esperado V(R), e se o N(S) é menor então o pacote é descartado, indicando que já foi recebido anteriormente. Se forem iguais, o pacote é aceito. Se o N(S) é maior que o V(R), então é solicitada a retransmissão dos pacotes faltantes entre o pacote esperado V(R) e o pacote que chegou N(S). O comando de rejeição seletiva (SREJ) é utilizado quando somente um pacote está faltando, e o comando de rejeição (REJ) é utilizado quando dois ou mais pacotes devem ser retransmitidos.

Em qualquer momento a ER destino pode entrar em estado de "ocupada temporariamente" se algum problema de sobrecarga nas filas de recebimento de pacotes ocorrer ou a aplicação do usuário não responder. Neste caso, um pacote contendo uma resposta de ER ocupada (DM) é enviado para a ER origem, que envia pacotes esporadicamente solicitando informações de estado (UI) da ER destino até que esta condição desapareça, iniciando a retransmissão dos pacotes pendentes.

Se o ack é recebido pela ER origem, a janela deslizante é inicializada e novos pacotes podem ser enviados.

Se os pacotes enviados pela ER origem não receberem um ack dentro de um tempo especificado T1, ocorre time-out e os pacotes pendentes são retransmitidos. Isto ocorre por um número pré-determinado de vezes, indicado pelo parâmetro *número máximo de tentativas*. Não ocorrendo sucesso nesta tentativa de solução de erros, a ER origem entra em estado de "desconexão" (DM).

Em qualquer momento que uma ER estiver no estado de "transferência de dados", ela pode solicitar a desconexão e encerramento da comunicação, através do envio de um comando DISC. A ER que receber este comando deve responder com um pacote de reconhecimento de desconexão (UA) e entrar no estado de "desconexão", como ilustra a figura 4.6, na parte "desconexão". Recebendo a resposta UA, o outro ER também entra no estado citado.

No estado de "desconexão", uma ER monitora os pacotes recebidos e reage ao recebimento de uma solicitação de conexão (SABM ou SABME), como já citado. Ao receber um pacote de solicitação de informação (UI), responde com um pacote de sistema desconectado (DM). Os fluxogramas do *Apêndice B* detalham os procedimentos de comunicação entre duas estações descritos acima.

O exemplo da figura 4.7 ilustra a comunicação realizada entre a ER1 e a ER2, utilizando duas estações repetidoras (R1 e R2) repassando os pacotes (P1, P2, P3,... P*n*), uma vez que a ER1 e a ER2 não estão próximas o suficiente para trocarem informações diretamente.

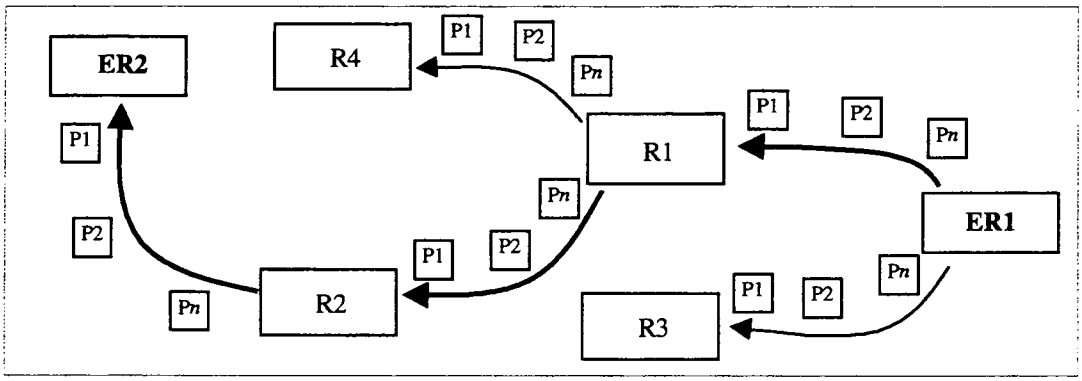


Figura 4.7 – Simulação de comunicação entre estações de rádio, utilizando repetidoras.

A ER1 disponibiliza os pacotes para o meio de transmissão, que vão para todas as direções (*broadcast*). Todas as estações repetidoras próximas captam estes pacotes, entretanto somente a que estiver especificada no próximo campo "Repetidora" destes pacotes é que os retransmitirão (as outras repetidoras os descartam). No exemplo, as estações repetidoras R3 e R4 não estão na lista de repetidoras, o que faz com que não retransmitam os pacotes recebidos, desprezando-os. Nos pacotes retransmitidos, o bit que indica "*passou por esta*

repetidora" é alterado e campo FCS é recalculado. Este procedimento acontece para todas as repetidoras e todos os pacotes até chegarem na estação destino.

A figura 4.8 representa a estrutura de um dos pacotes de informação que estão sendo transmitidos pela ER1.

Flag	Endereço Destino	Endereço Origem	Endereço Repetidora	Endereço Repetidora	PID	Controle	Dados	FCS	Flag
01111110	**ER2**	**ER1**	**R1**	**R2**	-	**I, N(S), N(R)**	*	*	**01111110**

Figura 4.8 – Exemplo de um pacote de informação do RPWNT utilizado no exemplo da figura 3.7.

4.8 Considerações finais

Este capítulo descreveu a especificação do protocolo para rádio pacote RPWNT. Foram apresentadas as principais características deste protocolo, os tipos de pacotes propostos e suas finalidades, as estruturas destes pacotes e seus campos, os pacotes de comandos e pacotes de respostas, os parâmetros deste protocolo, os procedimentos de tratamento de erros e o controle da numeração dos pacotes. Por fim, foram apresentados os procedimentos envolvidos na comunicação entre duas estações de rádio e um exemplo de comunicação entre estas estações.

O protocolo AX.25, que possui uso específico para ambiente em rádio pacote e considerado um padrão mundial pela comunidade de rádios amadores, foi tomado como base para o protocolo proposto. As diferenças entre o protocolo RPWNT e o protocolo AX.25 são mostradas no quadro 4.9.

ITEM	AX.25	RPWNT
Camadas do modelo OSI	Física e enlace	Rede e transporte
Verificação do meio físico antes de transmitir pacotes	Sim	Não
Independência do hardware da placa de rádio	Não	Sim
Meios de transmissão que podem ser utilizados	Ondas de rádio	Ondas de rádio e cabos

Quadro 4.9 - Diferenças entre os protocolos RPWNT e AX.25.

É importante observar que foram poucas as alterações realizadas na especificação do protocolo RPWNT em relação ao protocolo base AX.25, e nenhuma alteração interfere em sua funcionalidade.

Algumas decisões na definição do protocolo foram tomadas, de modo a manter a compatibilidade entre estes dois protocolos:

- O campo PID da estrutura do pacote de informação foi mantido na especificação do protocolo RPWNT. Este campo indica qual é o protocolo que está sendo usado na camada de rede. Uma vez que o protocolo AX.25 trabalha nas camadas física e enlace e o protocolo RPWNT trabalha nas camadas de rede e transporte, este campo se torna desnecessário;
- O bit "C" do subcampo da identificação secundária (SSID), refere-se às versões anteriores do protocolo AX.25 e não é utilizado pelo protocolo RPWNT.

É importante observar que a modularização do código fonte, situando-o nas camadas de rede e transporte, torna o protocolo RPWNT independente do hardware utilizado, sendo portável para diversos meios de transmissão, como ondas de rádio e cabos.

A utilização de soquetes para a implementação do protocolo RPWNT no ambiente Windows NT, como mencionado na seção 2.3, permite que códigos fontes de aplicações de usuários que utilizam outros protocolos de transporte possam ser reaproveitados, devendo sofrer poucas alterações, como endereçamento envolvido na comunicação e chamadas para as funções de soquetes.

CAPÍTULO 5

UM PROTÓTIPO DO DRIVER DE TRANSPORTE

Este capítulo apresenta um protótipo do protocolo RPWNT com o objetivo de verificar a aplicabilidade do protocolo proposto ao ambiente Windows NT ao qual ele se destina, seguindo os padrões da arquitetura de rede deste sistema operacional. A implementação do protótipo permite a observação e discussão da experiência e detecção de problemas e dificuldades desta adaptação.

As alterações realizadas na especificação do protocolo RPWNT em relação ao protocolo base AX.25 não interferem em sua funcionalidade. Com isto, a verificação das funcionalidades do protocolo RPWNT não é objetivo deste protótipo, somente a sua adaptabilidade no ambiente Windows NT.

A seção 5.1 apresentada o contexto e limites do protótipo na implementação do protocolo RPWNT no ambiente Windows NT, a seção 5.2 descreve o fluxo de dados de uma requisição da aplicação do usuário até a transmissão para outro computador, a seção 5.3 apresenta as estruturas e funcionalidades dos objetos implementados e a seção 5.4 descreve os testes realizados com este protótipo. As experiências e dificuldades percebidas na adaptação do protótipo são discutidas nas considerações finais do capítulo.

5.1 Contextualização

O protocolo RPWNT propõe uma alternativa de comunicação de rádio pacote para transmitir dados entre computadores usando a tecnologia de rádio amador, sendo específico para o ambiente de rede Windows NT.

Em sua especificação, no capítulo 4, é proposto que o protocolo RPWNT seja um driver de transporte, situado entre o driver do sistema de arquivos e o driver da adaptadora de rede, como descrito na arquitetura de rede do Windows NT, no capítulo 2, e melhor ilustrado na

figura 2.1 - por outros driver não nativos. O Windows NT permite que drivers de transporte não nativos possam ser adaptados em sua arquitetura de rede [Ddk96a] [Stu95]. O objetivo do protótipo é observar a aplicabilidade do protocolo proposto neste ambiente.

Como apresentado na seção 2.4, para a adaptação do protocolo RPWNT ao ambiente de rede Windows NT, é necessário o desenvolvimento de um driver de transporte e uma biblioteca auxiliadora de soquetes (WSH). Eles permitem que uma aplicação do usuário envie mensagens através da rede e que uma aplicação situada em outro computador receba e reconheça estas mensagens. Logo, os objetos do protótipo são: driver de transporte, biblioteca auxiliadora de soquetes e aplicação do usuário. A localização destes objetos no Windows NT é visualizada na figura 2.3 do capítulo 2. Alguns requisitos necessários para a implementação deste protótipo são citados no *Apêndice C.*

5.2 Fluxo de dados da aplicação do usuário ao meio físico de transmissão

Os objetos do protótipo interagem entre si e com o ambiente através de mensagens de requisição de serviços. O fluxo de dados 5.1 ilustra o envio de uma mensagem da aplicação do usuário até sua transmissão na forma de pacotes pela adaptadora de rádio para outro computador.

A aplicação do usuário realiza chamadas para rotinas genéricas (API's) que fazem uso de soquetes. Através destas rotinas de soquetes, a aplicação indica qual é a família de endereços que está sendo utilizada para identificar os computadores envolvidos na comunicação, o tipo de comunicação, que pode ser orientada a conexão ou sem conexão, e o número que identifica qual protocolo é usado. Todas as solicitações da aplicação que necessitam interagir com o sistema operacional ou com qualquer dispositivo de hardware, conversam com o subsistema Win32, que é responsável por prover uma interface única de comunicação entre o sistema operacional e as aplicações que utilizam diversos ambientes. Maiores detalhes sobre as funcionalidades do subsistema Win32 estão descritos no *Apêndice A.*

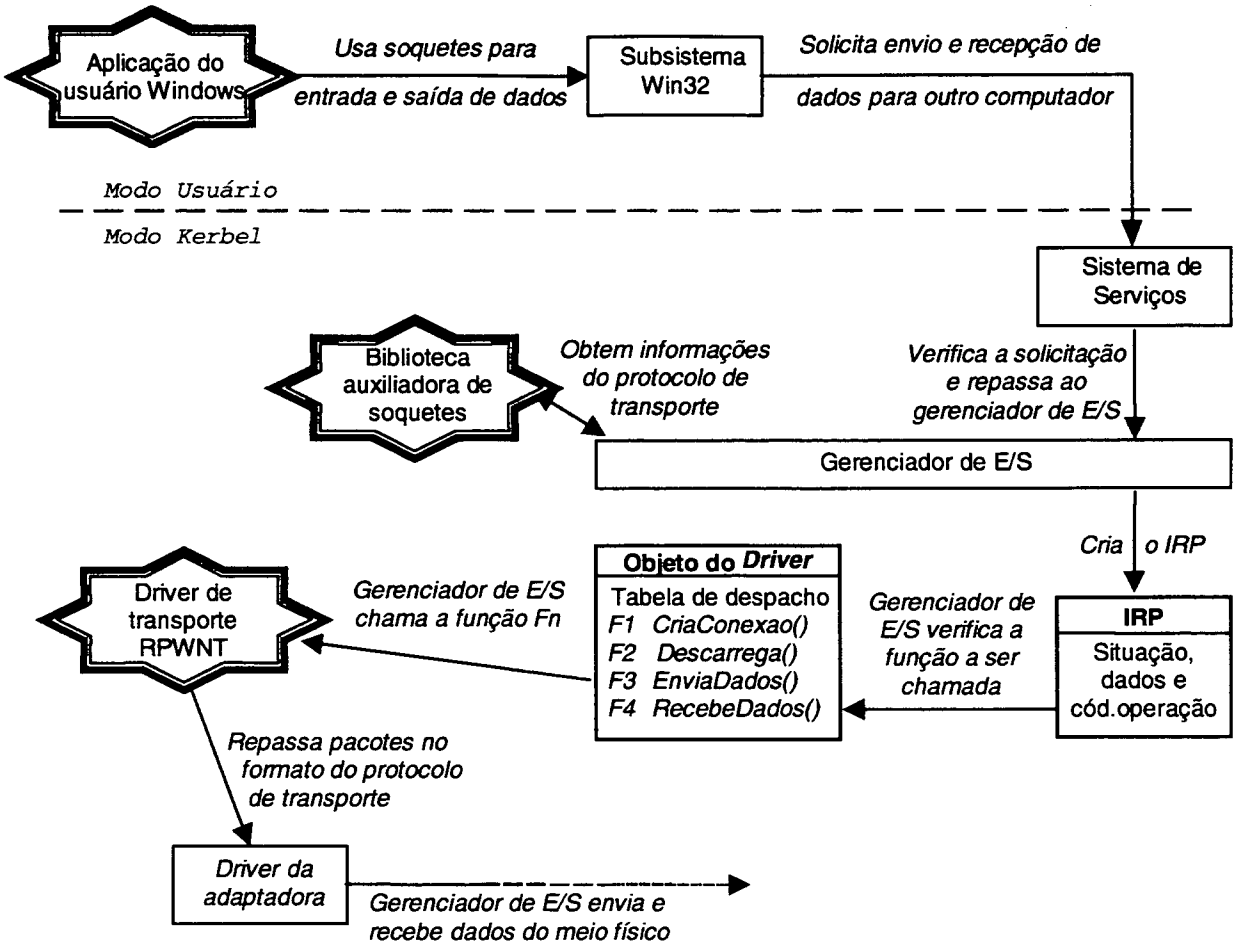


Figura 5.1 – Exemplo do fluxo de envio de dados de uma aplicação do usuário para a rede.

Tanto a aplicação do usuário quanto o subsistema Win32 são executados em modo usuário e este último solicita ao sistema operacional a realização de tarefas em modo kernel. Isto garante a portabilidade da aplicação para diferentes arquiteturas de CPU.

O "*Sistema de Serviços*", que é um componente do sistema operacional, trabalha em modo kernel e recebe a solicitação do subsistema Win32, acionando o gerenciador de entrada e saída para manipular a solicitação de envio de dados da aplicação. Este gerenciador é responsável por repassar as solicitações aos diferentes tipos de drivers de dispositivos e controlar todas as operações relacionadas a esta requisição [MM96].

Usando o provedor de serviços de soquetes, o gerenciador de entrada e saída descobre para qual protocolo de transporte deve ir a solicitação de envio de dados, e usa a biblioteca

auxiliadora de soquetes específica para obter informações e realizar tratamentos de dados para este protocolo.

O gerenciador de entrada e saída cria um pacote de requisição IRP (*I/O Request Packet*), que é uma estrutura de dados que contém informações de como a operação de entrada e saída é processada. O Windows NT possui uma lista de tarefas padrões que podem ser solicitadas aos seus drivers. A identificação da tarefa solicitada e a referência aos dados que devem ser enviados para o outro computador são armazenada no pacote IRP.

Quando um driver de transporte é iniciado, sendo carregado para a memória, o gerenciador de entrada e saída cria uma estrutura de dados que contém informações sobre este driver (*Objeto do Driver*), como por exemplo os dispositivos que estão ligados a ele e as rotinas específicas deste driver que poderão ser chamadas nas diversas ocasiões do funcionamento do protocolo. Estas rotinas são armazenadas numa tabela conhecida como Tabela de Despacho. Mais informações sobre os objetos que são disponibilizados pelo Windows NT e suas funcionalidades são apresentados no *Apêndice D.*

O gerenciador de entrada e saída de posse do pacote IRP, verifica qual é a operação solicitada pela aplicação do usuário e busca a rotina específica na estrutura do objeto do driver, fazendo a chamada para esta função.

Depois dos pacotes de dados serem criados no formato do protocolo proposto, o gerenciador de entrada e saída aciona o driver da adaptadora de rede que está ligado ao protocolo de transporte, para enviá-los ao meio físico de transmissão.

O driver de transporte se comunica com o gerenciador de entrada e saída, para enviar e receber dados da aplicação do usuário, através de chamadas às rotinas e uso das estruturas de dados definidas na biblioteca e camada limítrofe TDI, e usa a biblioteca e camada limítrofe NDIS para enviar e receber dados do driver da adaptadora de rede.

Quando pacotes chegam ao computador destino, são recebidos pelo driver da adaptadora de rede, que envia estes dados para o driver de transporte. O gerenciador de entrada e saída verifica qual é a função indicada pelo driver de transporte para tratar pacotes vindos do driver da adaptadora de rede e realiza a sua chamada, fornecendo estes pacotes.

Depois de tratados pelo protocolo de transporte, um pacote de resposta é enviado ao computador origem, e os dados recebidos são enviados para a aplicação do usuário, que deve estar em modo de espera, preparada para receber os dados.

Quando o computador origem recebe o pacote de resposta que estava sendo aguardado, o processo inverso ao mostrado na figura 5.1 é iniciado pelo driver do dispositivo. A figura 2.2 ilustra o recebimento de pacotes provenientes do meio físico. O driver de transporte armazena a situação final da operação solicitada e a resposta no IRP pendente e manda-o de volta para o gerenciador de entrada e saída, que responde a situação final da operação à aplicação do usuário.

5.3 Estrutura do driver de transporte

Este seção apresenta o funcionamento interno do driver de transporte desenvolvido neste protótipo. O driver de transporte é o componente do ambiente de rede Windows NT responsável pela implementação de um protocolo de transporte. No contexto definido para este protótipo na seção 5.1, que é de adaptar o protocolo no ambiente de rede Windows NT, o driver de transporte tem a função de empacotar e repassar os dados recebidos da aplicação do usuário para o driver da adaptadora de rede, e vice-versa. Não são realizados controle de fluxo e tratamento de erros, especificados no capítulo 4. Os pacotes de dados gerados por este driver de transporte e repassados para o driver da adaptadora de rede são enviados por broadcast, sem a necessidade de uma conexão entre os computadores. A implementação do driver de transporte permite observar a adaptabilidade do protocolo no ambiente de rede Windows NT.

Neste sistema operacional, uma das grandes diferenças entre um driver e uma aplicação é a estrutura de controle. As aplicações são executadas do início ao fim sob o controle de uma função chamada *Main* ou *WinMain*, que determina qual a seqüência em que várias subrotinas são chamadas. Um driver em modo kernel, ao contrário, possui um conjunto de rotinas que são chamadas quando necessário pelo gerenciador de entrada e saída. As rotinas do protótipo são chamadas nos seguintes momentos: quando o driver é carregado ou descarregado (rotinas de inicialização e finalização), quando uma aplicação de usuário faz uma solicitação (rotinas de despacho) e quando o driver da adaptadora de rede solicita alguma operação ou repassa

pacotes provenientes do meio físico (rotinas de ligação com o NDIS). A figura 4.2 ilustra as principais funções do driver de transporte do protótipo.

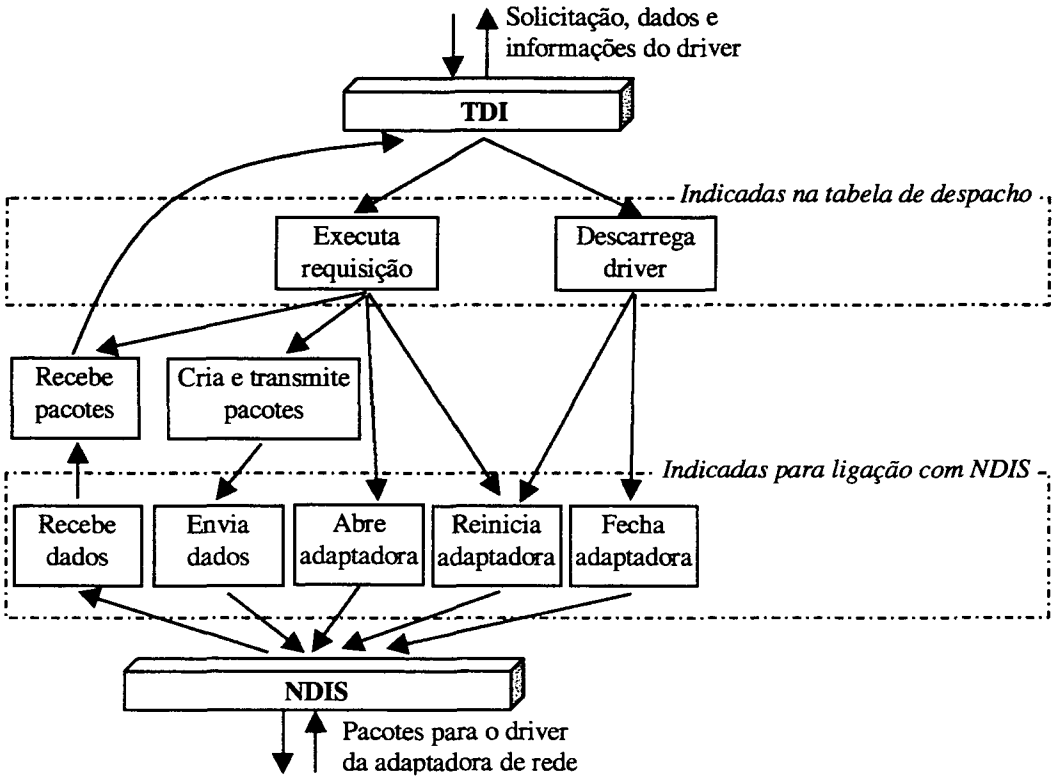


Figura 5.2 – Exemplo da estrutura das principais rotinas do driver de transporte do protótipo.

5.3.1 Rotinas de inicialização e finalização

Quando o sistema operacional carrega o driver de transporte para a memória, o gerenciador de entrada e saída cria um objeto que contém informações sobre este driver. Este gerenciador obtém o caminho da biblioteca auxiliadora de soquetes para este protocolo , que é o diretório onde está localizada esta biblioteca, as famílias de endereços e tipos de pacotes que o driver de transporte suporta. O gerenciador de entrada e saída executa a função inicial obrigatória do driver, a *DriverEntry*, que tem como objetivo principal informar ao gerenciador de entrada e saída quais são os tipos de operações suportadas por este driver [Tom97] e quais funções poderão ser chamadas quando algum evento for solicitado pela aplicação do usuário ou pelo sistema operacional.

Este gerenciador lê as informações de configuração do protocolo que estão armazenadas no registro do sistema (*Registry*) e faz o registro do driver para que ele seja reconhecido pelo sistema operacional como um "protocolo de transporte". Em uma estrutura de dados padrão da biblioteca NDIS, são indicadas as rotinas que serão executadas pelo gerenciador de entrada e saída para fazer a ligação do driver de transporte com esta biblioteca. Estas rotinas são apresentadas na subseção a seguir.

A função DriverEntry informa ao sistema quais são as rotinas deste driver que podem ser chamadas pelo gerenciador de entrada e saída quando uma solicitação é realizada pela aplicação do usuário. As rotinas fazem parte da tabela de despacho do objeto do driver e a subseção 5.3.3 apresenta estas rotinas.

Para cada adaptadora de rede ligada ao protocolo de transporte, é criado um Objeto do Dispositivo, que é uma estrutura que contém informações sobre o estado e características destas adaptadoras.

Depois do driver iniciado, ele fica sem execução até ser acionado pelo gerenciador de entrada e saída com uma solicitação da aplicação do usuário ou com um pacote proveniente da adaptadora de rede. Sempre que o driver de transporte for carregado para a memória do computador, ele permanecerá lá até que uma reinicialização do sistema ocorra. Para que o driver seja retirado da memória uma função é definida na rotina DriverEntry e indicada na tabela de despacho do objeto do driver. Através de uma solicitação do usuário ou da execução da Aplicação de Rede do Painel de Controle (NCPA – *Network Control Panel Application*), esta função destrói todos os objetos criados e libera a memória alocada pelas estruturas de dados em uso.

5.3.2 Rotinas de ligação com o NDIS

A biblioteca NDIS utiliza uma estrutura de dados pré-definida pelo Windows NT para fazer a ligação entre o driver de transporte e o driver do dispositivo. Toda a troca de mensagens entre estes drivers é realizada através de chamadas de rotinas que são indicadas nos campos desta estrutura, sendo citadas a seguir. Algumas destas rotinas são ilustradas na figura 5.2.

- *OpenAdapterCompleteHandler* indica se a abertura de uma adaptadora de rede foi completada com sucesso pelo driver do dispositivo;
- *CloseAdapterCompleteHandler* indica se o fechamento de uma adaptadora de rede foi completada com sucesso pelo driver do dispositivo;
- *ResetCompleteHandler* indica se a reinicialização de uma adaptadora de rede foi completada com sucesso pelo driver do dispositivo;
- *RequestCompleteHandler* indica se a requisição solicitada ao driver do dispositivo foi completada com sucesso pela biblioteca NDIS;
- *SendCompleteHandler* cria pacotes no formato NDIS, completa com dados e envia para o dispositivo, não esperando as confirmações destes pacotes enviados, sendo livres de conexão;
- *TransferDataCompleteHandler* recebe a indicação que a transferência de dados foi completada. Responde ao gerenciador de entrada e saída liberando os IRP's pendentes e as estruturas de dados usadas pelos pacotes enviados. Incrementa os contadores de referência e prepara as configurações para a próxima transferência, chamando a rotina de envio novamente;
- *ReceiveCompleteHandler* indica a recepção de um pacote vindo do meio físico de comunicação. Este pacote é livre de conexão, ou seja, não necessita de confirmação de pacotes.

Maiores detalhes de como devem ser implementadas as rotinas que fazem a ligação com a biblioteca NDIS são encontrados em [Ddk96a, Ddk96b].

5.3.3 Rotinas de despacho

Estas são as rotinas que fazem parte da tabela de despacho do objeto de dispositivo, e que são responsáveis por executar tarefas passadas pelo gerenciador de entrada e saída quando algum evento é solicitado pela aplicação do usuário ou pelo sistema operacional. As rotinas definidas para o driver de transporte do protótipo são apresentadas a seguir.

- O gerenciador de entrada e saída faz solicitações internas de operações a serem realizadas pelo driver. Estas operações podem ser solicitações de usuários transformadas para o modo kernel, solicitações do sistema operacional durante a execução do driver de transporte e solicitações repassadas pelo driver do dispositivo. Algumas das ações realizadas pela rotina *ExecutaRequisicaoKernel* são: a transmissão de dados utilizando pacotes que não são orientados à conexão, como broadcast; o recebimento de requisições do computador remoto, armazenando-as para execução posterior; e o recebimento dos dados vindos do computador remoto através de pacotes que não são orientados à conexão.
- *ExecutaRequisicaoUsuario* recebe as solicitações dos usuários e as transformam em códigos internos do Windows NT e então faz a chamada para a função *ExecutaRequisicaoKernel*, para a execução da solicitação em modo kernel;

Maiores detalhes de como devem ser implementadas as rotinas da tabela de despacho são encontrados em [Ddk96a, Ddk96b], e alguns conceitos que melhor esclarecem o processo de desenvolvimento de um driver de transporte no Windows NT são apresentados no *Apêndice E.*

5.4 Estrutura da biblioteca auxiliadora de soquetes

A biblioteca auxiliadora de soquetes (WSH - *Windows Sockets Helper*) resolve as ambigüidades que podem ocorrer entre diferentes protocolos em uso. A aplicação do usuário realiza chamadas para os soquetes através das rotinas da biblioteca Win32. O soquete utilizado especifica uma família de endereços, um tipo de soquete e um protocolo. Estes três argumentos identificam unicamente um driver de transporte suportado pelo soquete.

O provedor de serviços de soquetes (MSAFD) verifica a igualdade entre os argumentos fornecidos e as informações padrões de configuração da biblioteca auxiliadora de soquetes, armazenadas no registro do sistema. Encontrando esta igualdade, chama a função interna específica que é fornecida pela biblioteca auxiliadora de soquetes. Se não houver igualdade, ocorre uma falha na chamada da aplicação que utiliza o soquete.

Através desta biblioteca auxiliadora, o gerenciador de entrada e saída identifica quais as funções do driver de transporte que utilizam a camada limítrofe TDI que devem ser chamadas.

As rotinas desenvolvidas para a biblioteca auxiliadora de soquetes do protótipo são apresentadas a seguir.

- *WSHEnumProtocols* : retorna a lista de protocolos que a biblioteca WSH suporta;
- *WSHGetSockAddrType* : obtém o tipo do endereço a ser utilizado pelo soquete ;
- *WSHGetSocketInformation* : recupera as informações do soquete armazenadas em uma estrutura de dados padrão do Windows NT;
- *WSHGetWildCardSockAddr* : retorna a máscara que indica o formato do endereço do soquete;
- *WSHGetWinsockMapping* : retorna informações sobre a família de endereços, tipo de soquete e protocolo suportado pela biblioteca WSH;
- *WSHNotify* : é chamada para notificar a transição do estado do soquete, como abertura, fechamento, e outros;
- *WSHOpenSocket2* : abre um soquete, verificando a validade dos parâmetros família de endereços, tipo de soquete e protocolo;
- *WSHSetSocketInformation* : altera as informações que são armazenadas na estrutura de dados de configuração do soquete;
- *WSHGetWSAProtocolInfo* : retorna o ponteiro para as informações deste protocolo, que estão armazenadas na estrutura de dados de configurações do soquete;
- *WSHAddressToString* : retorna uma cadeia de caracteres que representa o endereço do soquete que pode ser usado com propósitos de exibição em tela;
- *WSHGetBroadcastSockAddr* : obtém um endereço válido para o soquete, com objetivo de envio de dados para toda parte, por broadcast;
- *WSHStringToAddress* : converte um cadeia de caracteres para um endereço aceito pelo soquete;

Maiores detalhes sobre possíveis funcionalidade das rotinas da biblioteca auxiliadora de soquetes são encontrados em [Ddk96a].

5.5 Aplicação do usuário

Para que o objetivo de enviar mensagens em forma de pacotes pudesse ser alcançado, foi desenvolvida uma aplicação para ser executada no ambiente do Windows NT, que simula a funcionalidade do aplicativo "ping", utilizado pelo protocolo TCP/IP.

A aplicação abre um soquete usando as informações da família de endereços, o tipo de comunicação e o número do protocolo de transporte do protótipo registrado no Windows NT e faz a associação do endereço local do computador com o soquete. Configura o tipo de comunicação a ser utilizada como broadcast. Faz a alocação de memória para os buffers de recebimento e de envio de pacotes. Envia um pacote e depois entra em estado de espera para receber a resposta e outros pacotes. Recebendo um pacote, envia outro com o mesmo endereço de origem e buffer de dados recebidos. Dois computadores ficam trocando pacotes infinitamente, até que uma das aplicações interrompa a seqüência com o uso das teclas "Ctrl+C". Todas as chamadas de funções e valores de retorno são mostrados na tela. A maioria das rotinas que utilizam soquetes gera uma chamada para as rotinas internas da biblioteca auxiliadora de soquetes.

Rotinas que devem ser chamadas em uma aplicação de usuário para utilização de soquetes são encontradas em [Msd99, Sdk96].

5.6 Testes realizados

Para a observação da aplicabilidade do driver de transporte do protótipo no ambiente Windows NT, e não a verificação das funcionalidades do protocolo RPWNT, testes foram realizados nas diversas etapas do desenvolvimento. Foram testadas as funções de inicialização e finalização do driver de transporte; a instalação do driver no ambiente de rede do Windows NT; o carregamento e descarregamento do driver na memória; a associação do driver de

transporte com os drivers de dispositivo; a associação do driver com a aplicação e a biblioteca auxiliadora de soquetes, desenvolvidas neste protótipo; e, enfim, a transmissão e o recebimento de mensagens entre dois computadores. A seguir são apresentadas algumas observações a respeito destas atividades.

Com as funções de inicialização e finalização do driver de transporte concluídas, este pôde ser criado através da compilação e linkedição, certificando que o ambiente de desenvolvimento funciona. Este processo de criação do driver é apresentado no *Apêndice F.*

Para a instalação do driver exemplo no ambiente de rede Windows NT, foi criado um arquivo que contém todas as informações necessárias para que este processo ocorra automaticamente, sem a intervenção manual do usuário. Este arquivo padrão, de nome *oemsetup.inf*, é utilizado pela Aplicação de Rede do Painel de Controle (NCPA). Os passos para a instalação do driver de transporte são apresentados no *Apêndice G.*

O teste de carregamento automático do driver para a memória gerenciada pelo ambiente Windows NT foi realizado com a reinicialização do computador, e o teste de descarregamento da memória foi realizado pelo aplicativo NCPA.

Para verificar se a ligação do driver de transporte com a adaptadora de rede existe, foi utilizado o aplicativo WINOBJ, fornecido pelo conjunto para desenvolvimento de softwares (SDK – *Software Development Kit*) da empresa Microsoft Corporation. Com esta ligação, percebe-se que os objetos internos do driver exemplo foram criados com sucesso.

Com a biblioteca WSH pronta, um programa que abre um soquete e obtém informações do protocolo foi utilizado para verificar as chamadas das rotinas desta biblioteca.

A aplicação do usuário foi desenvolvida e testada inicialmente para o uso do protocolo IPX/SPX e depois de pronta foi adaptada para o protocolo exemplo do protótipo. Esta adaptação foi realizada com pequenas alterações na passagem dos parâmetros das rotinas que usam os soquetes e na definição da estrutura de dados do endereço do computador.

Por último, o objetivo de enviar pacotes de aplicações de usuário situadas em dois computadores ligados entre si foi alcançado, verificando-se a aplicabilidade do protocolo de transporte no ambiente de rede Windows NT.

Os procedimentos para a realização de depurações do código fonte do driver de transporte são citados no *Apêndice H.*

5.7 Considerações finais

Este capítulo descreveu o protótipo do protocolo de transporte RPWNT para o ambiente de rede Windows NT. Este protótipo utiliza comunicação do tipo broadcast, enviando pacotes e recebendo confirmações através de uma aplicação de usuário criada, atuando como o aplicativo "ping" do protocolo TCP/IP. Para a ligação entre esta aplicação e o protocolo de transporte, foi implementada uma biblioteca auxiliadora de soquetes. Em todas as fases da implementação deste protótipo, os testes de funcionamento apresentaram sucesso.

Depois da contextualização deste protocolo no ambiente citado, o fluxo interno de dados da aplicação até meio físico de transmissão foi descrito, sendo apresentados os objetos internos do Windows NT necessários neste fluxo. O funcionamento interno do driver de transporte foi mostrado, bem como suas rotinas de inicialização, finalização, de ligação com a biblioteca NDIS e da tabela de despacho. A biblioteca auxiliadora de soquetes foi desenvolvida para fazer o intercâmbio entre as rotinas genéricas de soquetes e as específicas do driver de transporte.

A implementação do protótipo permitiu a observação e discussão da experiência e detecção de problemas e dificuldades desta adaptação. Os parágrafos abaixo indicam as dificuldades encontradas na implementação do protótipo do driver de transporte, para que estas possam ser minimizadas ou evitadas em implementações futuras.

- Difícil acesso à documentação. Por exemplo, não foram encontradas documentações distribuídas livremente, sem custo e que tratassem do desenvolvimento de drivers de transporte para este sistema operacional;

- Erros e divergências entre documentações do sistema operacional Windows NT. Além disto, documentações para desenvolvimento de drivers incompletas e sem organização coerente;

- Difícil acesso às ferramentas de desenvolvimento. Isto se deve às restrições impostas pelo fabricante do sistema operacional que limita o desenvolvimento de drivers ao uso das próprias ferramentas, que são de alto custo. Em consulta ao mercado brasileiro, realizada em 31/01/2000, o valor dos softwares necessários para o desenvolvimento de um driver de transporte era :
  - Windows NT Workstation versão 4.0, ao preço de R$ 595,00 e Windows NT Server versão 4.0, ao preço de R$ 3.805,00;
  - Visual C++ Enterprise versão 6.0, ao valor de R$ 2023,00;
  - Assinatura anual do MSDN (*Microsoft Developer Network*) Professional, que disponibiliza as bibliotecas DDK (*Device Driver Kit*) e SDK (*Software Development Kit*), ao valor de R$ 1597,00, e a versão Universal ao valor de R$ 6.434,00.

- Alto número e complexidade de funções e estruturas de dados das bibliotecas de interface de drivers e sistema operacional. Por exemplo, para as camadas NDIS e TDI, são disponibilizadas 1470 rotinas de acesso e 240 estruturas de dados que, em geral, são compartilhadas por diferentes funções e objetos. Chegou-se a detectar estruturas de dados com 80 campos;

- Impossibilidade da utilização do programa exemplo de um driver de transporte fornecido pelo fabricante do sistema operacional Windows NT, devido aos seguintes fatores :
  - Não é fornecida a biblioteca auxiliadora de soquetes para este driver de transporte, impossibilitando a ligação entre as funções do driver e as API's de aplicações;
  - Nenhuma documentação do driver de transporte exemplo é fornecida e os comentários nas linhas de programa não possuem clareza de entendimento.

- Impossibilidade de uso de diversos recursos do software de desenvolvimento, como por exemplo, a interface gráfica, a compilação e linkedição interativa, a ajuda *on-line* (help) e o uso do depurador passo-a-passo (debug) para detecção de erros de execução. Como o driver de transporte trabalha em modo kernel, este último procedimento causa travamentos no sistema operacional. O apêndice H descreve o difícil procedimento de depuração de drivers de transporte.

CAPÍTULO 6

CONCLUSÃO

A principal contribuição deste trabalho foi a experimentação do protocolo para rádio pacote RPWNT a ser utilizado no ambiente de rede Windows NT.

Este trabalho apresentou a arquitetura interna de rede do Windows NT, comparando seus componentes com as camadas do modelo OSI. Destacou-se a importância do driver de transporte, da biblioteca auxiliadora de soquetes e das camadas limítrofes TDI e NDIS, para a adaptação de protocolos de transporte para o sistema operacional Windows NT.

Para se conseguir a independência da adaptadora de rede e poder ser utilizado em diversos meios de transmissão, o código de tratamento do protocolo proposto deve ser isolado em um módulo que trabalhe nas camadas de rede e transporte do modelo OSI, sendo definido como um protocolo de transporte. No entanto, para a sua implementação, dois módulos devem ser fornecidos pelo desenvolvedor, que são o driver de transporte e a biblioteca auxiliadora de soquetes. As camadas limítrofes são importantes por serem a interface de comunicação entre o protocolo de transporte e o resto do sistema operacional.

O protocolo RPWNT foi baseado no protocolo AX.25. A decisão do protocolo base foi tomada após estudos teóricos realizados com protocolos e tecnologias para comunicação em redes sem fio. Este texto apresentou as principais características das tecnologias estudadas e suas vantagens e desvantagens em relação ao uso em ambiente de rádio pacote.

Como o protocolo AX.25 é considerado o padrão mundial para rádio pacote pela comunidade de rádio amadores, buscou-se a total compatibilidade entre as especificações deste protocolo e o RPWNT. É importante observar que foram realizadas poucas alterações na especificação do protocolo RPWNT em relação ao protocolo base AX.25. As alterações na especificação do protocolo RPWNT não interferem em sua funcionalidade e foram feitas para garantir sua adaptabilidade ao ambiente de rede Windows NT como um driver de transporte. Para preservar a compatibilidade entre as duas especificações, alguns campos foram mantidos

na especificação do protocolo RPWNT. Em relação às implementações, o protocolo AX.25 trabalha nas camadas física e de enlace, enquanto o protocolo RPWNT trabalha nas camadas de rede e transporte. Isto impede que os dois protocolos estabeleçam comunicação direta entre eles em uma rede.

Um protótipo do protocolo de transporte RPWNT para o ambiente de rede Windows NT foi desenvolvido e apresentado, permitindo a observação e discussão da experiência de adaptação de um protocolo neste ambiente. Este protocolo utiliza comunicação do tipo broadcast, enviando pacotes e recebendo confirmações através de uma aplicação de usuário. Para a ligação entre esta aplicação e o protocolo de transporte, foi implementada uma biblioteca auxiliadora de soquetes. Este protótipo foi verificado e os testes de funcionamento apresentaram sucesso.

Com este trabalho, foi possível a aquisição e sistematização do conhecimento de uma arquitetura que, apesar de muito utilizada, não é divulgada e é de difícil acesso. Ainda foram apresentadas neste texto, como resultado da experiência adquirida no desenvolvimento do protótipo, as dificuldades encontradas no estudo e implementação referentes ao ambiente Windows NT.

6.1 Perspectivas

O protocolo RPWNT, depois de implementado, será uma alternativa para disponibilizar a técnica de rádio pacote no ambiente Windows NT, proporcionando um aumento na utilização desta tecnologia e possibilitando a criação de novas aplicações com custos reduzidos. Alguns exemplos de aplicações que podem ser implementadas com a comunicação em rede sem fio de baixa velocidade são :

- Envio esporádico de informações de medições coletadas em lugares distantes, como florestas, rios, represas, montanhas, entre outros;
- Comunicação sem fio de computadores distantes entre si, como uma cooperativa agrícola e um usuário em seu sítio ou fazenda;

- Adaptação de aplicações que utilizam soquetes, como *chat*, correio eletrônico e até navegadores (*browsers*) usando páginas em hipertexto, permitindo que códigos fontes de aplicações de usuários possam ser reaproveitados.

Pode ser realizada a avaliação de desempenho do protocolo RPWNT em utilização nos vários ambientes em que é portável, bem como a avaliação de desempenho do protocolo RPWNT em comparação com outros protocolos que utilizam o ambiente de rádio pacote.

Para a comunicação direta do protocolo RPWNT com o protocolo AX.25 em uma rede, pode ser criado um driver de dispositivo para o Windows NT que repasse a estrutura do pacote do protocolo RPWNT para o meio físico de transmissão e deixe o controle do fluxo dos pacotes e tratamentos de erros para o protocolo proposto realizar. Como a arquitetura de rede do Windows NT é modular, alterações na especificação e implementação do protocolo RPWNT não são necessárias. Um exemplo de implementação de um driver de adaptadora de rádio é fornecido por Branco em [Bra00].

As tarefas de desenvolvimento de protocolos de transporte específicos podem ser facilitadas com o caminho inicial indicado pelo protótipo criado, e várias aplicações podem surgir com a utilização de protocolos de transporte neste ambiente. Alguns exemplos são os produtos analisadores de rede, que obtêm dados que trafegam no barramento e os filtros de pacotes, como os utilizados em *firewalls*. Isto é possível porque o driver de transporte pode ler e filtrar dados que passam por ele, além de conter as especificações dos protocolos de rede. Dois exemplos de produtos citados e disponíveis no mercado são encontrados em [Ber00, Mod00].

REFERÊNCIAS BIBLIOGRÁFICAS

[AJ95] AGHAZARM, Bruno, e JÚNIOR, Jedey A. M. **Transmissão de Dados em Sistemas de Comutação**. São Paulo : Érica. 1995.

[Bak96] BAKER, Art. **The Windows NT Device Driver Book : A Guide for Programmers**. 1ª. Edição. New Jersey, USA : Prentice-Hall, Inc. 1996.

[BDM95] BOISSEAU, Marc, DEMANGE, M. e MUNIER, J-M. **High Speed Networks**. West Sussex, England: Ed. John wiley & Suns Ltda. 2ª ed., 1995.

[Ber00] ____. **Analisadores de Protocolos**. Home-page da empresa BERKANA : http://www.berkana.com.br (acessado em 06/02/2000).

[BN97] BEECH, William A., e NIELSEN, Douglas E., e TAYLOR, Jack. **AX.25 Link Access Protocol for Amateur Packet Radio.** Ver. 11/11/1997, p: iii. 1997.

[Bra00] BRANCO, Parahuari S. **Implementação de uma rede sem fio de baixo custo**. Curitiba, 2000. Dissertação (Mestrado em Informática) - Setor de Ciências Exatas, Universidade Federal do Paraná.

[Cam94] CAMARÃO, Paulo C. Bhering. **Glossário de Informática**. Rio de Janeiro : LTC - Livros Técnicos e Científicos, 2ª edição, 1994.

[Cap79] CAPETANAKIS, John I. **Generalized TDMA: The multi-accessing tree protocol**. Tutorial: IEEE transactions on communications, vol. com-27, n° 10, october 1979.

[Che94] CHEN, Kwang-Cheng. **Médium Access Control of Wireless LANs for Modile Computing**. Primeira publicação em IEEE Network Magazine. Vol. 8, Num. 5. Set-Out/1994.

[CL99] CÂMARA, Daniel e LOUREIRO, Antônio A. F.. **Redes de computação móvel Ad Hoc**. Departamento de Ciência da Computação, Universidade Federal de Minas Gerais. Belo Horizonte – MG. 1999.

[Ddk96a] ____. Microsoft Windows NT Device Driver Kit (DDK) – **Network Drivers Book**, Content Version 4.0. Copyright Microsoft Corporation. Microsoft, Design Guide. July 1996.

[Ddk96b] ____. Microsoft Windows NT Device Driver Kit (DDK) – **Kernel Mode Drivers Book**. Content Version 4.0. Copyright Microsoft Corporation. July 1996.

[Ddk96c] ____. Microsoft Windows NT Device Driver Kit (DDK) – **Programmer's Guide Book**, *4.1 – Debugging Kernel Mode Drivers,* Content Version 4.0. Copyright Microsoft Corporation. July 1996.

[DN95] DESIMONE, Antonio e NANDA, Sanjiv. **Wireless data : System, standards, services**. Holmdel-USA : J. C. Baltzer AG - Science Publishers, p: 241-253. 1995.

[Fro96] FRONING, Andrew. Computação Móvel. **Revista Byte Brasil**. São Paulo : Rever Ltda., p: 19-46. Maio/1996.

[GB93] GASPARINI, Anteu F. L. e BARRELLA, Francisco Eugênio. **TCP/IP soluções para conectividade**. São Paulo: Editora Érica, 2ª ed., 1993.

[GPG98] GANZ, Aura, PHONPHOEM, Anan e GANZ, Zui. **Robust superpoll protocol for IEEE 802.11 Wireless Lan**. Amherst-USA : Multimedia Wireless Laboratory, ECE Departament, Universit of Massachusetts. p: 1-5. 1998.

[Gun97] GUNTER, Jost. **TCP/IP on FlexNet - Just Another Layer**. Baltimore, Maryland: 16ª Digital Communications Conference. American Radio Relay League & Tucson Amateur Packet Radio Corporation. 1997.

[I3e97] Standard IEEE 802.11, "**Wireless Lan**", P802.11, 1997.

[MH94] MYERS, Brian e HAMER, Eric. **Dominando a Programação no Windows NT.** 1ª. Edição. Rio de Janeiro: LTC – Livros Técnicos e Científicos Editora, p: 9-31. 1994.

[MM96] MACHADO, Francis B. e MAIA, Luiz Paulo. **Arquitetura de Sistemas Operacionais.** 2ª. Edição. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora S.A., p: 180-194, 1996.

[Mod00] ____.____. Home-page da empresa MÓDULO INFORMÁTICA : http://www.modulo.com.br (acessado em 06/02/2000).

[MSA86] MOURA, José Beltrão, SAUVÉ, Jacques Philippe e ARAÚJO, José Fábio Marinho. **Redes locais de computadores: protocolos de alto nível e avaliação de desempenho.** São Paulo: McGrawHill. 1986.

[Msd99] ____.____ MSDN 2000 - Help. **Microsoft Software Development Network** (MSDN). Version for Windows 2000. July 1999.

[Pre95] PRESSMAN, Roger S. **Engenharia de Software.** São Paulo : Makron Books. 3ª edição. 1995.

[RBP95] ROCHOL, Juergen, BARCELOS, Marcelo Boeira e PUFAL, Henrique. **Comunicação de dados em redes celulares de telefonia móvel (RCTM).** 13° SBRC. Belo Horizonte: Ed. José Marcos Silva Nogueira - UFMG, p: 247, 1995.

[Ren82] ____. **Especificações dos protocolos de acesso à RENPAC (X.3, X.28, X.29 e X.25).** SESA, DPS 25, versão 4, 1982.

[Rob78] ROBERTS, Laurence G. **The evolution of packet switching.** Tutorial: Proceeding of the IEEE, vol. 66, n° 11, november 1978.

[Sdk96] ____. **Microsoft Win32 Software Development Kit** (SDK) – Win32. Version for Windows NT 3.51 and Windows 95. July 1996.

[Sil95] SILVEIRA, Jorge Luís. **Comunicação de Dados e Sistemas de Teleprocessamento**. São Paulo : Makron Books. 1995.

[Siq96] SIQUEIRA, Ethevaldo. Tudo o que você precisa saber sobre CDMA. **Revista RNT**. São Paulo : Telepress, ano 18, nº 208A, Dezembro/1996.

[SLC97] SOARES, Luiz Fernando G., e LEMOS, Guido e COLCHER, Sérgio. **Das LANs, MANs e WANs às Redes ATM**. Rio de Janeiro: Campus. p: 413-415. 1997.

[Stu95] ____. Supporting Microsoft Windows NT 3.51 – **Student Book (Course 505)**. Release: 07/95. United States of America: Microsoft Corporation, p: 264-284. 1995.

[Tan94] TANENBAUM, Andrew. **Redes de Computadores**. 1ª. Edição – Tradução de: Computer Networks. Rio de Janeiro: Campus, p: 325-526. 1994.

[Tar97] TAROUCO, Liane M. Rockenbach. **Protocolo AX.25**. Home-page http://penta.ufrgs.br/tp951/cac_pro.html (acessado em 17/07/1997).

[Tom97] TOMLINSON, Paula. Sending IOCTLs to Windows NT Drivers. **Revista Windows Developer's Journal**. Boulder - USA : Miller Freeman Publication. Vol. 8, num. 2, p: 6-26. February 1997.

[Wat97] WATT, Steve. **Frequently Asked Questions (FAQs) for Digital Amateur Radio**. http://www.smartpages.com/faqs/radio/digital-faq/faq.html (Version 1.17, Rev. 22/08/1993, acessado em 20/08/1997).

[Zuc86] ZUCCHI, Wagner Luiz. **Transmissão de dados em redes de computadores**. Rio de Janeiro: LTC - Livros Técnicos e Científicos. 1986.

APÊNDICE A

AMBIENTE WINDOWS NT

O Windows NT foi escrito, na sua maior parte, em linguagem C. O código do sistema dependente do hardware foi isolado em uma biblioteca conhecida como HAL (*Hardware Abstraction Layer*). Essas características garantem ao Windows NT uma facilidade muito grande de utilizar variados tipos de dispositivos e ser portado para diferentes plataformas de hardware, como RISC ou CISC.

O Windows NT é estruturado de acordo com o modelo de sistemas operacionais cliente-servidor. Nessa arquitetura, o sistema é dividido em processos servidores, sendo cada um responsável por oferecer um conjunto de serviços, como de arquivos, criação de processos, de memória e escalonamento.

A arquitetura do sistema é baseada em objetos, que vêm a ser tipos de dados abstratos manipulados somente por um conjunto especial de serviços. Neste sistema operacional, qualquer recurso do sistema é representado por um objeto, como: processo, enlace (*thread*), seção, arquivo, porta, indicador de acesso *(token)*, evento, semáforo e temporizador. Mesmo não sendo um sistema totalmente orientado a objetos, o Windows NT utiliza objetos para representar qualquer recurso do sistema que possa ser compartilhado por mais de um processo. Com isto, o sistema tem acesso e manipula seus recursos de forma uniforme, inclusive o compartilhamento de recursos entre processos [MM96].

O Windows NT utiliza um modelo para descrever privilégios de acesso ao processador, para proteger o sistema operacional contra certas instruções que poderiam ocasionar sérios problemas à integridade e segurança do sistema. Estes modos de acesso são disponibilizados pelo hardware e a CPU deve habilitar estes modos do modelo de privilégios.

Em modo kernel, uma tarefa pode executar instruções privilegiadas, e tem acesso completo a qualquer dispositivo de entrada e saída. Em modo usuário, uma aplicação só pode executar instruções que não oferecem risco ao sistema.

O hardware impede a execução de instruções privilegiadas e executa checagem de acesso nas referências para memória e dispositivos de entrada e saída de dados. Quando uma aplicação necessita de um serviço que incorra em risco para o sistema, o Windows NT altera o modo de acesso do processador para o modo mais privilegiado. Ao término da rotina do sistema, o modo de acesso é retornado para o modo usuário e o controle é passado novamente para a aplicação.

A estrutura do Windows NT é dividida em duas partes : o subsistema protegido e o executivo, como mostrado na figura A1. Os processos servidores e os gerenciadores de tarefas desta estrutura são descritos nos parágrafos seguintes.

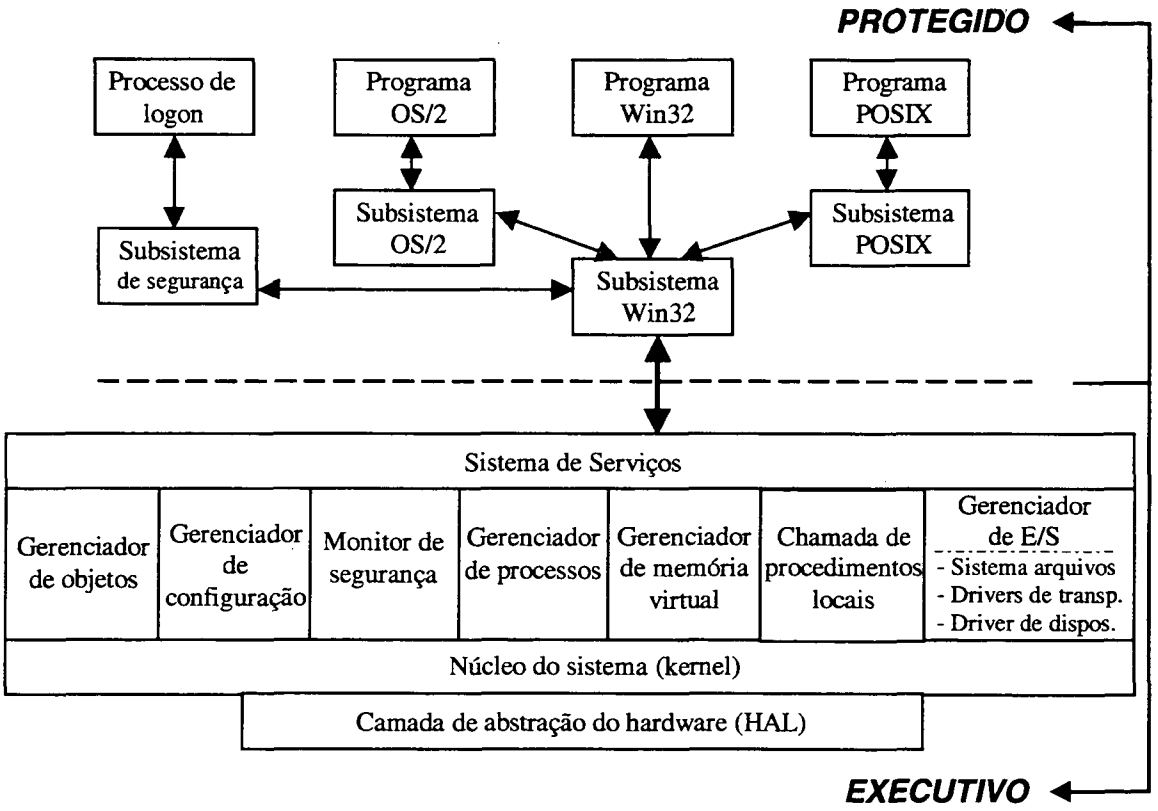


Figura A1 – Estrutura do sistema operacional Windows NT

O subsistema protegido é formado por processos servidores que oferecem serviços para uma aplicação de usuário ou outro servidor, como por exemplo o Subsistema Win32. Tem este nome porque cada servidor possui seu próprio espaço de memória virtual, sendo a comunicação entre eles feita mediante troca de mensagens. Este subsistema é executado em

modo usuário. O principal subsistema é o Win32, que fornece rotinas com objetivo de gerenciar a tela, como o gerenciamento de janelas e interfaces gráficas, e com ela todas as entradas relacionadas ao teclado e ao mouse. Aplicações de usuário executadas em outros ambientes operacionais utilizam o subsistema Win32 para interagir com o executivo do Windows NT. Como o Win32 controla todos os consoles, ele sempre sabe qual janela está sendo focalizada e qual subsistema deve receber a entrada do mouse e do teclado.

O executivo é o sistema operacional propriamente dito, que é executado em modo kernel e consiste de uma série de serviços de sistema. Quando uma aplicação cliente solicita algum serviço, ela executa uma chamada ao sistema através de funções de interface (API – *Application Program Interface*) específicas de um determinado subsistema. Por estar em modo kernel, , a aplicação ganha a vantagem de seu código origem ser portável para diferentes arquiteturas de CPU.

O executivo realiza a comunicação entre cliente e servidor através de um mecanismo de troca de mensagens, conhecido como Chamada de Procedimento Local (LPC), quando ambos estão situados na mesma máquina. O Windows NT também possui um recurso de mensagem mais geral para chamadas de procedimentos remotos (RPC) capaz de chamar servidores em máquinas remotas.

O kernel é a camada de software responsável por mecanismos relacionados à CPU. Faz o gerenciamento do processador, manipulação de interrupções e exceções, escalonamento e programação de enlaces, e sincronização de ambientes com múltiplos processadores.

A camada de abstração de hardware (HAL) é a camada mais baixa dos serviços executivos do modo kernel. Ela contém um conjunto de funções específicas da plataforma para manipular entrada e saída de baixo nível, interrupções, caches de hardware e a comunicação entre multiprocessadores. Gera um modelo abstrato de qualquer hardware que não faça parte da CPU.

O gerenciador de objetos é responsável pela criação e manutenção de todos os objetos de dados do sistema. Protege estes objetos contra acessos de outros componentes do sistema operacional não autorizados.

A principal tarefa do gerenciador de configuração é manter um modelo de todos os aplicativos e dispositivos instalados na máquina, bem como suas configurações. Para tanto, utiliza uma base de dados chamada registro do sistema (*Registry*) [Bak96], que armazena informações como nome do aplicativo ou dispositivo, fabricante, número de série, licença de uso, configurações, parâmetros, ligações com outros dispositivos, dependências de outros aplicativos, e muito mais. O registro é organizado em diretórios ou pastas, subdiretórios, chaves e valores.

Quando um enlace pede para usar um objeto, o gerenciador de objetos chama o monitor de referência de segurança para determinar se ele está autorizado a receber os direitos de acesso pedidos. A decisão depende do nível de autorização do enlace, armazenado em um *token* de acesso, e das restrições de acesso do objeto, armazenadas em sua lista de controle de acesso.

O gerenciador de memória organiza as alocações de memória para cada programa rodando no Windows NT, evitando que eles utilizem o mesmo espaço de endereços. A cada programa é dado seu próprio conjunto de endereços de memória, variando de 0 a 4 Gibabytes, e o gerenciador de memória transforma esses endereços virtuais imaginários nos endereços reconhecidos como físicos pelos programas, impedindo que os programas vejam ou alterem qualquer endereço físico em uso por outro programa. O gerenciador de memória também faz paginação.

O gerenciador de processos cria e destrói processos e seus enlaces, também suspendendo e retomando estes enlaces. Os recursos de multitarefa do Windows NT dependem da manipulação de processos e enlaces do sistema. [MH94].

O gerenciador de entrada e saída é responsável por receber solicitações de entrada e saída dos processos e repassá-las aos diferentes tipos de drivers de dispositivos de entrada e saída de dados, como teclados, impressoras, discos rígidos, discos flexíveis, cdroms, e mesmo solicitações para a rede. Cada solicitação de entrada e saída é encaminhada através de uma estrutura de dados (*I/O Request Packet* – IRP) que controla como estas operações são processadas [MM96]. O trabalho principal deste gerenciador é aceitar requisições de entrada e saída, usualmente de aplicações em modo usuário, criar IRPs que as representem, rotear os IRPs para os drivers apropriados do Windows NT, acompanhar estes pacotes até que sejam

completados e retornar a situação para o requisitante original das operações de entrada e saída.

Todos os drivers de dispositivos são construídos para suportar o modelo imposto por esta estrutura. Qualquer componente que transforme um pedido de entrada lógico em um pedido físico mais específico pode ser um driver, como os sistema de arquivos, drivers da adaptadora e drivers de transportes.

Para fazer melhor uso do tempo de CPU, o executivo implementa toda operação de entrada e saída em modo assíncrono, onde as funções que iniciam uma entrada e saída retomam a execução do programa antes que esta tarefa termine, podendo realizar outras operações enquanto as informações dos dispositivos não retornam à CPU [MH94].

APÊNDICE B

FLUXOGRAMAS DOS PROCEDIMENTOS DO RPWNT

Este apêndice tem o objetivo de ilustrar os procedimentos e tarefas executadas pelo protocolo RPWNT, e se dividem em: solicitação de conexão pela estação de rádio origem, recebimento da solicitação de conexão pela estação de rádio destino, envio de pacotes de dados pela estação de rádio origem, recebimento dos pacotes de dados pela estação de rádio destino e envio e recebimento de pacotes de dados pelas estações repetidoras.

B1. Convenções

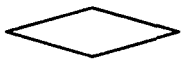
As convenções utilizadas nas ilustrações dos fluxogramas do protocolo RPWNT são:

Estado 

Recepção de Solicitação 

Geração de Resposta 

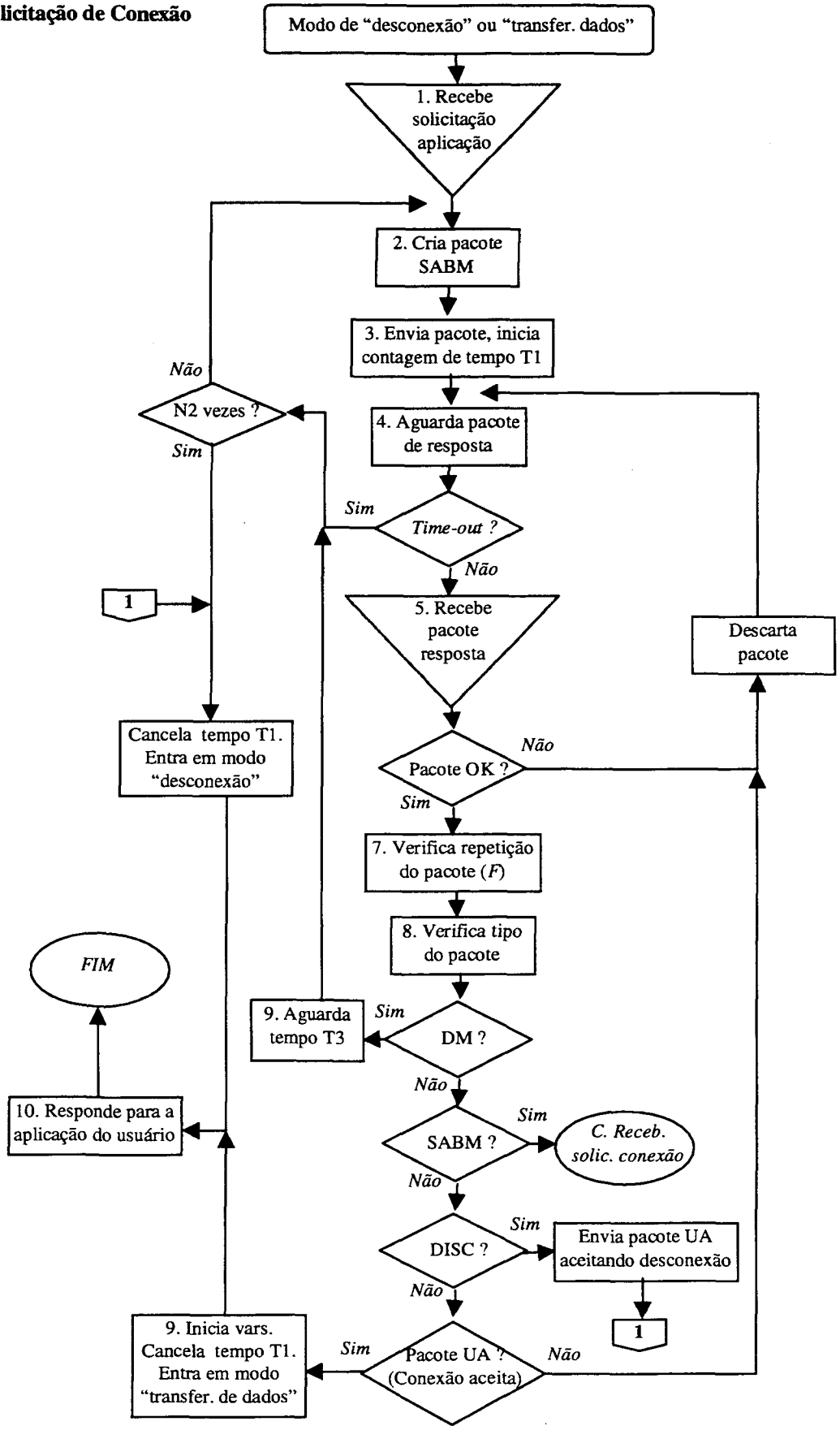
Descrição de Procedimento 

Decisão 

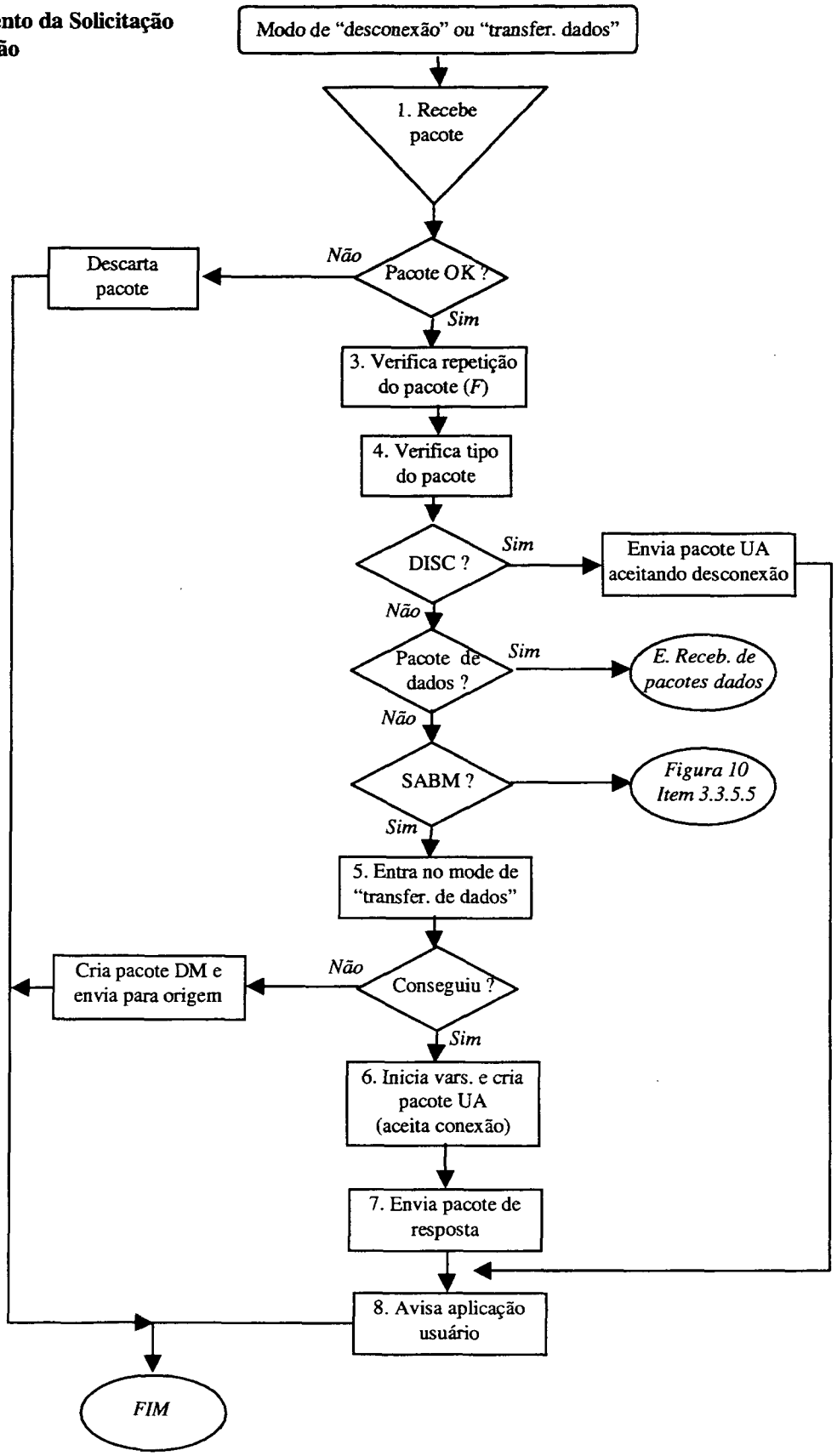
Fim ou outra descrição 

Ligação com outro ponto 

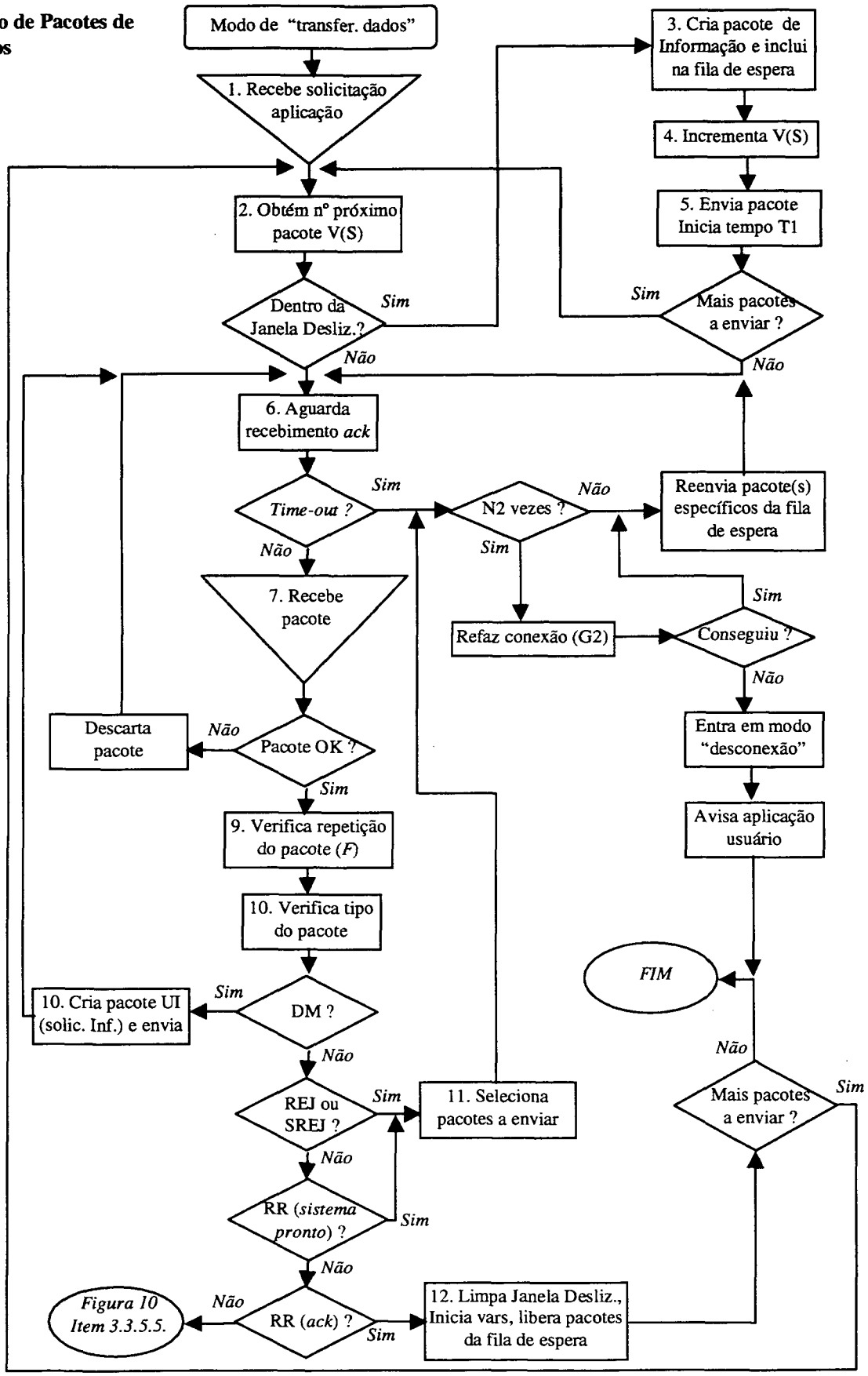
B2. Solicitação de Conexão



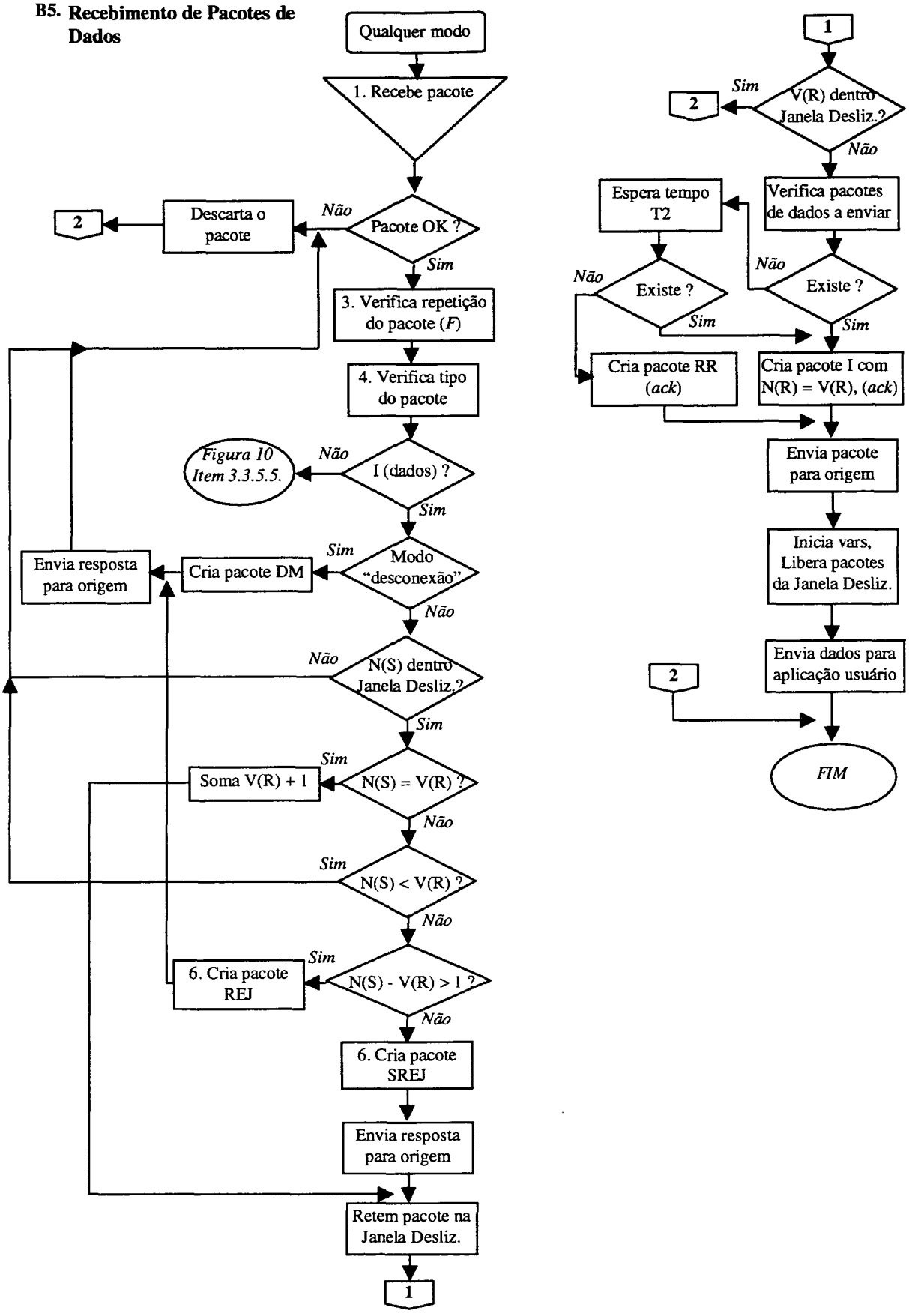
B3. Recebimento da Solicitação de Conexão



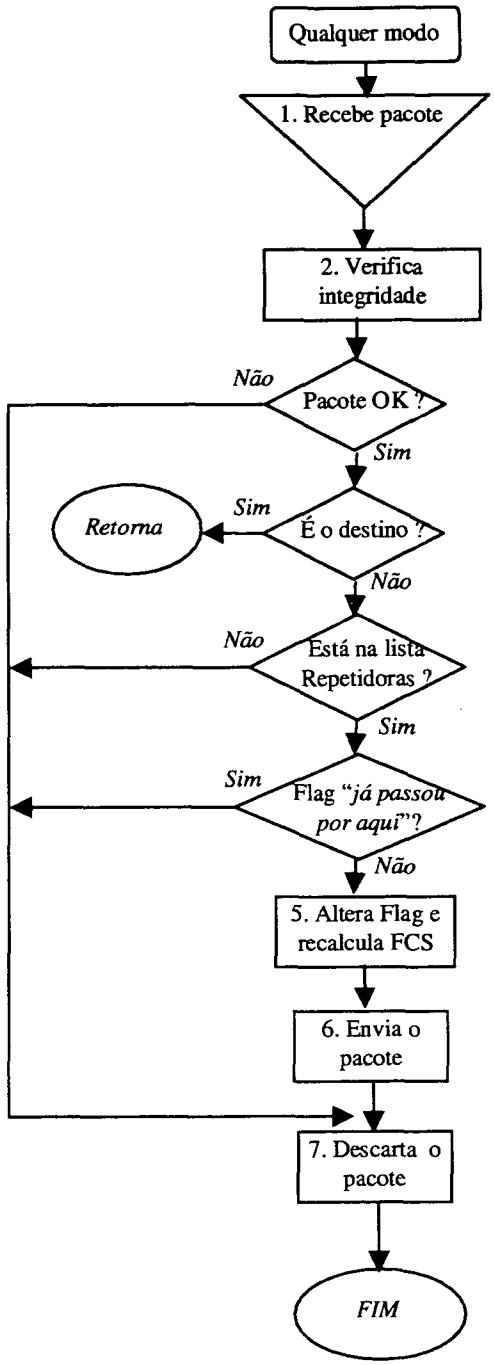
B4. Envio de Pacotes de Dados



B5. Recebimento de Pacotes de Dados



B6. Recebimento e Envio de Pacotes por Estações Repeditoras



APÊNDICE C

REQUISITOS PARA A IMPLEMENTAÇÃO DO PROTÓTIPO

Para o desenvolvimento do protótipo proposto no ambiente de rede Windows NT, são requeridos softwares específicos de desenvolvimento disponibilizados unicamente pela empresa Microsoft Corporation [Ddk96a]. Os pacotes de softwares necessários para o desenvolvimento do protótipo são listados abaixo :

a) Software de desenvolvimento Visual C++ versão 5 ou superior, para compilação e linkedição do driver;

b) Biblioteca WIN32 SDK (*Software Development Kit*) da Microsoft, para fornecer API's para soquetes. É pré-requisito para a instalação do DDK;

c) Biblioteca DDK (*Device Driver Kit*) da Microsoft, para fornecer as funções e estruturas de dados internas do Windows NT para o desenvolvimento do driver, como por exemplo, a NDIS e a TDI;

No sistema operacional Windows NT já instalado, deve-se possuir privilégios de administrador do sistema para que softwares requeridos e o próprio driver possam ser instalados com sucesso;

APÊNDICE D

OBJETOS DO WINDOWS NT UTILIZADOS POR DRIVERS

IRP - Pacote de Requisição de Entrada e Saída

O gerenciador de entrada e saída é o responsável por receber solicitações de entrada e saída dos processos, como as aplicações de usuário, e repassá-las aos diferentes tipos de drivers de dispositivos. Ele é orientado a pacotes, ou seja, cada solicitação de entrada e saída é encaminhada através de uma estrutura de dados (*I/O Request Packet* – IRP) que controla como estas operações são processadas. Este gerenciador aloca a estrutura do IRP em uma região de memória do sistema que não é paginada. Os IRPs do driver de transporte podem ser repassados para os drivers da adaptadora, se necessário.

O IRP é dividido em 2 partes, sendo mostrados os principais campos :

- Cabeçalho – indica *"em que estágio se encontra a operação de entrada e saída"*. Os campos utilizados pelo driver de transporte são :
  - *IoStatus :* contém a situação final da requisição da operação de entrada e saída;
  - *MdlAddress :* ponteiro para a lista de endereços (*list descriptor*) onde estão indicadas as páginas de memória que contém o buffer do usuário;
  - *UserBuffer* : endereço do buffer do usuário;
- Pilha de Localização (*I/O stack location*) – indica *"o que o driver deve fazer"*. Os principais campos utilizados pelo driver são :
  - *MajorFunction :* especifica a operação solicitada pela aplicação do usuário que será executada pelo driver.
  - *Parameters :* são os parâmetros passados para as funções indicadas;

- *DeviceObject :* ponteiro para a estrutura que contém informações do protocolo de transporte (Objeto do Dispositivo, descrito mais adiante).

Objeto do Driver

Quando o driver é carregado para a memória, o gerenciador de entrada e saída cria uma estrutura que contém informações próprias deste driver, como os dispositivos que estão ligados e que serão manipulados por ele. Esta estrutura é chamada de objeto do driver (*driver object*). Neste processo de inicialização, este objeto recebe os ponteiros para as diversas funções específicas deste driver, e guarda-os em campos internos que constituem a tabela de despacho.

Quando o gerenciador de entrada e saída recebe um IRP de uma aplicação do usuário, ele verifica qual é a operação solicitada e busca a função específica na estrutura do Objeto do Driver, fazendo a chamada para esta função do driver. Os principais campos desta estrutura são :

- *DriverStartIo :* ponteiro para a função de inicialização do driver;
- *DriverUnload :* ponteiro para a função de finalização do driver;
- *MajorFunction:* vários campos com ponteiros para as funções especificadas pelo código da operação a ser realizada;
- *DeviceObject* : ponteiro para a estrutura que contém informações do protocolo de transporte (Objeto do Dispositivo, descrito a seguir);

Objeto do Dispositivo

Existem ocasiões que tanto o gerenciador de entrada e saída quanto operações específicas do driver de transporte precisam saber a situação das adaptadoras de rede ligadas a este driver. O Objeto do dispositivo (*Device Object)* é uma estrutura que contém informações sobre o estado e características de cada dispositivo ligado ao protocolo de transporte, existindo uma estrutura para cada protocolo.

O gerenciador de entrada e saída mantém anexado ao objeto do dispositivo ponteiros para o IRP corrente e para uma fila de IRPs pendentes. O quadro abaixo mostra os principais campos desta estrutura:

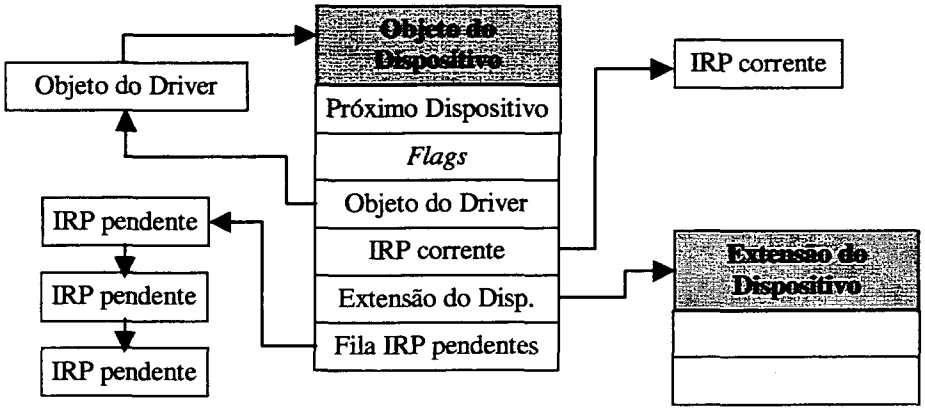


Figura D1 – Estrutura do Objeto do Dispositivo.

Extensões do Dispositivo

Utilizado para guardar informações particulares do driver de transporte, é simplesmente um bloco de memória não paginável que o gerenciador de entrada e saída anexa ao objeto do dispositivo. O desenvolvedor escolhe o tamanho e o conteúdo desta estrutura.

Não é aconselhável o uso de variáveis estáticas e globais na implementação do driver, devendo ser utilizada a Extensão do Dispositivo para que informações possam ser acessadas por outras funções internas.

Como esta estrutura é específica do driver de transporte, ela deve ser definida nos arquivos de cabeçalho deste.

Cita-se, para conhecimento, outros tipos de objetos disponibilizados pelo Windows NT e que não são utilizados no driver de transporte:

- *Objetos de controladoras* e *extensões de controladoras*, utilizados para desenvolvimento de driver de controladoras de disco rígido;

- *Objetos de adaptadoras*, para drivers de adaptadoras que geralmente usam mecanismos de Acesso Direto à Memória (DMA – *Direct Memory Access*). Normalmente adaptadoras de vídeo, algumas adaptadoras de rede e multimídia, entre outras.

- *Objetos de interrupções*, para drivers que causam interrupção de CPU, geralmente os de dispositivos físicos utilizam este recurso.

APÊNDICE E

CONCEITOS DO FUNCIONAMENTO DO WINDOWS NT

Alguns conceitos importantes do funcionamento do sistema operacional Windows NT que auxiliaram na implementação do driver de transporte proposto são apresentados a seguir.

Prioridades das tarefas da CPU

Diferentes arquiteturas de processadores possuem e manipulam diferentes prioridades de hardware. Para evitar estas dependências de arquiteturas, o Windows NT utiliza um esquema abstrato de prioridades chamado nível de interrupção de requisição (IRQL – *Interrupt Request Level*), que define o quanto é importante a tarefa que está sendo executada pela CPU no momento. Quanto maior o número indicado, maior a prioridade [Bak96]. O sistema operacional trabalha com os seguintes níveis :

- Nível de despacho (DISPATH_LEVEL), que permite adiamento de execuções de procedimentos e protela tarefas, trabalhando com memória não paginada;
- Nível de execução de procedimentos assíncronos (APC_LEVEL);
- Nível normal de execução de tarefas (PASSIVE_LEVEL), trabalhando com memória paginada.

Alocação de memória

Em situações específicas internas, o driver necessita alocar memória temporária. Para tanto, deve usar funções específicas do DDK, e não das bibliotecas padrões do C (como *malloc* e *free*) [Bak96]. Existem 3 opções :

- Pilha do kernel (*stack kernel*) : disponibiliza uma pequena porção limitada de memória não-paginada para variáveis locais durante a execução de rotinas específicas do driver;

- Pool paginado : memória em grandes quantidades da área paginada (páginas em disco) disponíveis para os drivers quando estão sendo executados em níveis de prioridade baixos (menor que DISPATCH_LEVEL);

- Pool não-paginado : drivers sendo executados em níveis de prioridades altas (maior ou igual a DISPATCH_LEVEL) podem usar bastante memória da área não-paginada. O sistema garante que a memória virtual é sempre da memória física, e não de disco (memória paginada).

Método de acesso à área de dados do usuário

Quando um programa de usuário faz uma requisição de entrada e saída, ele usualmente fornece o endereço da área de dados na memória (*buffer*). O Windows NT disponibiliza 2 métodos de acesso aos buffers dos usuários. Quando o driver é inicializado, ele chama o gerenciador de entrada e saída indicando o método que ele planeja usar [Bak96]. Os métodos são:

- Entradas e saídas armazenadas *(BIO – Buffered I/O)* : na saída de dados, o gerenciador de entrada e saída copia o conteúdo do buffer do usuário para a área do sistema antes de disponibilizar para o driver. Para requisições de entrada de dados, o driver preenche a área do sistema com dados vindos do dispositivo, e o gerenciador de entrada e saída copia de volta para o buffer do usuário e finaliza a operação. Há uma duplicação dos dados, em ambos os casos.

- Entradas e saídas diretas *(DIO – Direct I/O)* : o gerenciador de entrada e saída disponibiliza o endereço físico da página de memória onde está localizado o buffer do usuário, em uma lista de endereços (*List Descriptor*) para acesso direto pelo driver, não necessitando de duplicação dos dados. O gerenciador de entrada e saída prende o buffer do usuário inteiro na memória para prevenir falhas, até que a operação de entrada e saída tenha sido concluída.

O driver de transporte RPWNT utiliza o método de entradas e saídas diretas, como a maioria dos drivers de transporte, pois as adaptadoras de rede possuem canais rápidos de

comunicação com a memória e que transmitem grandes quantidades de dados. O método de entradas e saídas armazenadas não é indicado nestes casos, pois a cópia de dados entre os buffers na memória torna o procedimento lento.

Semáforos

Em ambientes com multiprocessadores, os dados compartilhados estão sujeitos a corrupção e erros. Para prevenir que isto aconteça, o Windows NT usa um procedimento de sincronização de objetos em modo kernel chamado semáforo (*Spin Lock*).

É simplesmente um objeto de exclusão mútua que se associa a um grupo específico de estrutura de dados. Quando uma porção de código em modo kernel quer alterar qualquer dado de uma estrutura guardada pelo procedimento, ele deve primeiro requerer um semáforo. Sendo que somente uma CPU possui o semáforo por vez, a estrutura de dados está livre de colisões. Qualquer CPU que requer um semáforo já alocado a outra CPU, deve esperar até que este seja liberado para continuar o processamento. A figura a seguir demonstra o funcionamento do semáforo [Bak96].

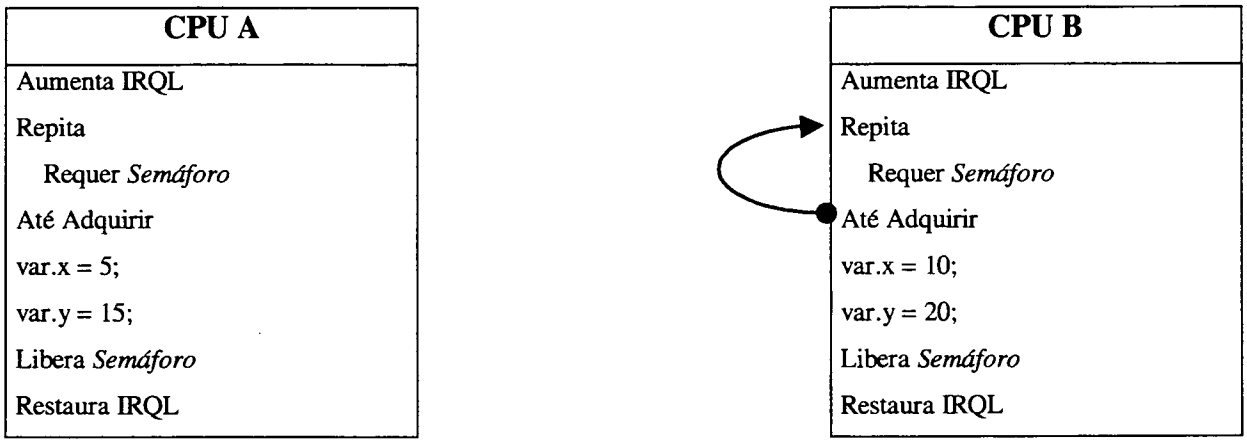


Figura E1 – Sincronização de múltiplas CPUs utilizando Semáforos.

APÊNDICE F

COMPILAÇÃO E LINKEDIÇÃO DO DRIVER DE TRANSPORTE

Um difícil aspecto no desenvolvimento de drivers para Windows NT é a necessidade de manter versões separadas para cada plataforma de hardware suportadas, porque são necessárias diferentes configurações para o compilador e linkeditor. O utilitário BUILD isola o desenvolvimento do driver das dependências de plataforma. Elaborado para trabalhar com o NMAKE, utilitário usado para compilar códigos fontes, usa um conjunto de palavras-chaves para descrever operações que devem ser executadas na construção do arquivo final do driver (*RPWNT.SYS*). O BUILD utiliza os arquivos de comandos MAKEFILE.DEF, MAKEFILE.PLT, I386MK.INK, ALPHAMK.INC, MIPSMK.INC e PPCMK.INC, que estão localizados no diretório \DDK\INC.

Especifica diferentes diretórios para Intel, MIPS, Alpha e PowerPC indicando onde será criado o arquivo do driver final, com o objetivo de auxiliar os projetos multiplataformas. A plataforma utilizada no desenvolvimento do driver RPWNT é a Intel. Também usa diretórios separados para as versões **Testada** (*checked*) e **Liberada** (*free*). A Testada contém informações de testes internos (*debug*) e usa símbolos DBG permitindo inclusão de códigos condicionais no driver. A Liberada é a versão otimizada, limpa, de menor tamanho e mais rápida, que deve ser usada como versão final deste driver. O quadro abaixo torna clara a idéia.

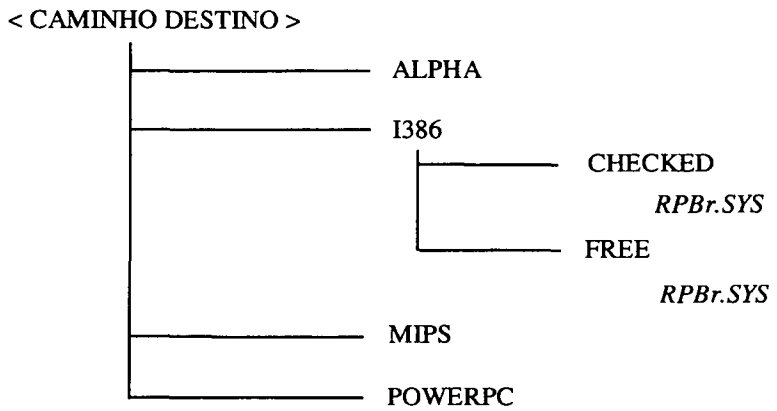


Figura F1 – Estrutura de diretórios dos produtos do BUILD

Os passos a seguir devem ser executados para a construção do driver, sendo que os números 1 a 3 só devem ser executados na primeira tentativa de construção, tornando-se desnecessários após isto :

1) Criar um arquivo chamado SOURCES no diretório onde se encontram os arquivos fontes do driver. Este arquivo identifica os componentes que serão utilizados para a criação da versão final do driver [Ddk96a];

2) No mesmo diretório, deve ser criado um arquivo chamado MAKEFILE (sem extensão e do tipo ASCII) necessário para criar qualquer driver com o utilitário BUILD. Contém somente a linha abaixo :

```
!INCLUDE $(NTMAKEENV)\MAKEFILE.DEF
```

3) Criar a estrutura de diretórios especificada na *figura F1*, de acordo com a plataforma de hardware sendo utilizada;

4) Configurar as variáveis de ambiente para que contenham as indicações dos diretórios onde estão instalados o DDK, SDK e VISUAL C++, nesta ordem. São as seguintes :
   - PATH = Diretórios dos arquivos executáveis (*.EXE);
   - LIB = Diretórios dos arquivos que contém as bibliotecas de funções (*.LIB);
   - INCLUDE = diretórios dos arquivos de cabeçalhos (*.H).

5) Ir ao diretório onde está localizado o arquivo SOURCES, usando o comando CD (*change directory do DOS*) para tanto;

6) Executar o comando BUILD para criar a versão do driver executável.

Como complemento da tela de saída, que mostra informações de como ocorreu o processo de criação do driver, o utilitário BUILD gera os seguintes arquivos de eventos no mesmo diretório onde está localizado o arquivo SOURCES :

- BUILD.LOG : lista os comandos invocados pelo NMAKE;
- BUILD.WRN : contém qualquer mensagem de aviso gerada durante o processo de construção;
- BUILD.ERR : contém a lista de erros gerada durante o processo de construção.

Como os drivers de transporte são executados em modo kernel, mensagens de aviso em tela ou em impressora não podem ser enviadas e a execução passo a passo das linhas do código fonte não podem ser realizadas, dificultando a depuração do driver. Com o objetivo de se fazer reconhecer as causas de erros e realizar o monitoramento do driver ou de determinados pedaços do código, são descritos no *Apêndice H* como são realizados depurações para drivers em modo kernel.

APÊNDICE G

INSTALAÇÃO DO DRIVER DE TRANSPORTE NO AMBIENTE WINDOWS NT

O Windows NT fornece um aplicativo que gerencia a instalação e ligação de componentes de rede, chamado de Aplicação de Rede do Painel de Controle (NCPA – *Network Control Panel Application*).

Quando o NCPA é usado para instalar um componente de rede, ele solicita o arquivo OEMSETUP.INF específico do driver que localiza os arquivos necessários, configura as chaves no Registro do Sistema, copia o RPWNT.SYS para o diretório DRIVERS, faz a ligação do protocolo com os componentes e reinicializa o sistema, carregando o driver automaticamente.

Os próximos itens descrevem como são realizadas as instalações do driver de transporte, nas formas automática e manual.

Instalação automática

Os arquivos INF fornecem um método automático para instalação de drivers. Para o caso de drivers de transporte, ele localiza os arquivos necessários, configura as chaves no Registro do Sistema, copia o RPWNT.SYS para o diretório DRIVERS, faz a ligação do protocolo com os componentes e reinicializa o sistema, carregando o driver automaticamente

As seções internas obrigatórias deste arquivo, que serve como pontos de entrada do programa de configuração (NCPA – *Network Control Panel Application*), são :

- *Seção de Identificação* : a seção **[Identify]** é a primeira a ser executada pelo NCPA. Retorna informações sobre o nome do driver está sendo instalado e seu tipo.

- *Seção de Opções de Retorno* : se a seção [Identify] indicar que o arquivo INF suporta o driver especificado, o NCPA chama a seção **[ReturnOptions]**, que recebe como parâmetro a indicação da linguagem para ser usada pelas janelas de dialogo. O arquivo INF disponibiliza variedades de linguagens. Esta seção retorna ao programa de configuração a lista de drivers que podem ser instalados por este arquivo INF (que no caso, é somente o RPWNT).

- *Seção de Opções de Instalação* : se a seção anterior retornar a situação de sucesso, indicando que a linguagem é suportada, NCPA mostra uma janela para o usuário escolher qual o driver a ser instalado. Se o usuário prosseguir com a instalação, a seção **[InstallOption]** é executada, recebendo como parâmetros a linguagem em uso, uma identificação da operação (*OptionID*), e o diretório onde está localizado o arquivo executável do driver (RPWNT.SYS).

- *Seção de Descrições dos Meios de Acesso dos Arquivos Origem* : consiste de uma linha para cada meio de acesso onde podem ser encontrados os arquivos para a distribuição do driver. Estes meios de acesso podem ser unidades acionadoras de disquetes, cdroms, ou disco rígido local (não precisando ser especificado). O formato de cada linha é :

  ChaveLinha = [*DescriçãoDisco*, TAGFILE = *LegendaArquivo*]

  - *ChaveLinha* : um número inteiro de 1 a 999 que identifica a linha que será acessada pelo mais tarde;
  - *DescriçãoArquivo* : nome do dispositivo que será mostrado ao usuário para que ele possa fazer a escolha;
  - *LegendaArquivo* : especifica o nome da legenda (*label*) do dispositivo que o programa de configuração irá solicitar para a inserção do disco correto onde estão os arquivos necessários. Por exemplo :

    [Source Media Descriptions]

    1 = "*Driver* Disk #1" , TAGFILE = disk1

    2 = "*Driver* Disk #2" , TAGFILE = disk2

O DDK possui um arquivo INF de exemplo que está localizado em \DDK\SRC\NETWORK\TDI. Possui uma boa definição das necessidades para a instalação de qualquer driver de transporte.

Maiores detalhes de como criar um arquivo INF são apresentados em [Ddk96a].

Instalação manual

Com o driver executável pronto, a instalação no ambiente do sistema operacional ocorre através dos seguintes passos :

1) Copiar o arquivo RPWNT.SYS gerado na construção, que se encontra no diretório <Caminho Destino>\i386\CHECKED, para o diretório %SystemRoot%\ \SYSTEM32\DRIVERS. O *Caminho Destino* foi especificado no arquivo SOURCES. A variável *SystemRoot* indica onde estão instalados os arquivos do sistema operacional (por exemplo : \WINNT);

2) Adicionar entradas apropriadas no registro do sistema usando o aplicativo REGEDT32, fornecido pelo Windows NT. Estas entradas são indicadas adiante pela *figura G1;*

3) Reinicializar o computador para que o sistema operacional fique ciente da existência do novo driver. A carga deste driver deve ocorrer automaticamente.

Se for detectada a existência de algum erro que não interrompa ou não ocasione problemas para o sistema operacional (erros não fatais), o driver pode ser recarregado para a memória sem a necessidade de reinicialização do computador. Deve ser utilizado o aplicativo "Dispositivos" (*Devices*) no "Painel de Controle" (*Control Panel*) deste sistema operacional para parar o driver. Então, deve ser copiado o driver correto novamente para o diretório *\DRIVERS* (passo 1) e o inicializar através do mesmo aplicativo. É necessário que a função para descarga do driver esteja especificada.

As principais entradas para o Registro do Sistema estão demonstradas a seguir. Com a definição destas chaves e valores, o driver passa a estar apto a funcionar.

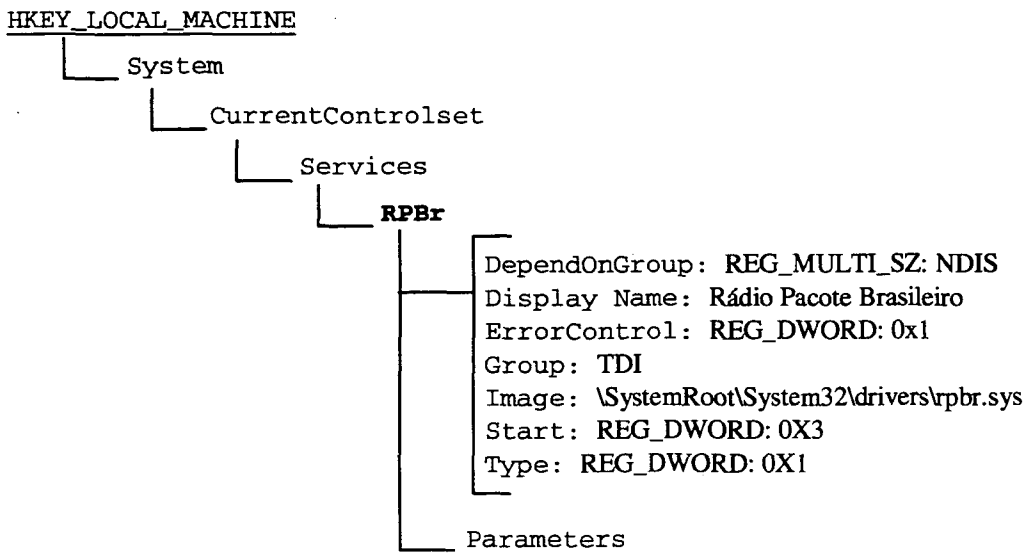


Figura G1 – Estrutura das chaves do Registro do Sistema para o driver *RPWNT.*

- *RPWNT* *: é a chave de serviço do driver. Todos os produtos que estiverem especificados abaixo da chave "*Services*", são tratados como serviços pelo Windows NT, podendo inclusive ser carregados e descarregados pela opção *Devices* (Dispositivos) do utilitário *Control Panel* (Painel de Controle);

- *DisplayName* : é o nome completo do driver que aparecerá nas janelas de configuração do Windows NT;

- *Image* : indica a localização e o nome do arquivo executável do driver;

- *Type* *: é o tipo do driver, que pode ser :
  - 0x1 : Modo Kernel ;
  - 0x2 : Sistema de Arquivos;

- *Start* *: indica quando é feita a carga do driver. O sistema operacional é inicializado em várias etapas e drivers podem ser carregados em qualquer uma delas, de acordo com suas funcionalidades e dependências com outros drivers. As opções podem ser :

* Chave obrigatória.

- 0x0 : iniciado pelo Carregador do Sistema Operacional (*service boot start*);
- 0x1 : após o sistema operacional ter sido carregado, mas enquanto está inicializando seus dispositivos internos (*service system start*);
- 0x2 : após o sistema operacional inteiro ter sido carregado e inicializado. O driver é carregado pelo Gerenciador de Controle de Serviços (*service auto start*). O protocolo RPWNT é iniciado aqui;
- 0x3 : deve ser inicializado manualmente, através da opção *Devices* (Dispositivos) do aplicativo *Controle Panel* (Painel de Controle);
- 0x4 : drivers não podem ser inicializados. Este valor deve ser alterado para que isto seja possível;

- *ErrorControl* *: o que o sistema operacional deve fazer quando um erro no driver ocorrer. As opções são :
  - 0x0 : registra o evento e ignora;
  - 0x1 : registra o evento e coloca em uma janela de mensagens. O driver RPWNT utiliza esta opção, apesar destas janelas não serem mostradas durante a execução de drivers;
  - 0x2 : registra o evento e reinicializa o computador, utilizando o última configuração com sucesso;
  - 0x3 : registra o evento e falha, se a última configuração com sucesso já foi utilizada;

- *Group* : é o grupo de drivers que o RPWNT pertence. É utilizado para especificar a seqüência de carga dos mesmos, visto que vários drivers podem estar configurados para inicializarem no mesmo estágio. Gera dependência na inicialização do driver. O grupo do RPWNT é o TDI, por ser um protocolo de transporte;

- *DependOnGroup* : indica qual grupo deve ser carregado antes do RPWNT ser inicializado. No caso, todos os drivers do grupo NDIS devem ser carregados antes dos drivers do grupo TDI.

- *Parameters* : é a chave que contém valores específicos de configuração do driver de transporte, por exemplo : NumMaxConexoes, TamJanelaDeslizante, TimeOutReconhecimento, TimeOutPacoteInterno, TimeOutLinkInativo, entre outros.

Em nenhuma das chaves acima foi indicada qual adaptadora do dispositivo o driver de transporte está ligado. Este protocolo pode se ligar a mais de uma adaptadora de dispositivo ao mesmo tempo, por exemplo : adaptadora de rede ou de rádio. O Windows NT possui várias regras internas específicas para a ligação entre componentes de rede, e vários outras chaves e valores devem ser especificados no Registro do Sistema para formar a hierarquia lógica que define as dependências deste componentes. Exemplos de chaves :

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RPWNT\ Linkage
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\RPWNT\CurrentVersion\ NetRules

Mais informações sobre o a instalação de drivers são descritos em [Ddk96b, Bak96].

APÊNDICE H

DEPURAÇÕES DO DRIVER DE TRANSPORTE

O primeiro teste a ser realizado é para verificar se a compilação e linkedição ocorreram com sucesso. Para tanto, as rotinas de inicialização e finalização do driver devem estar concluídas.

Depois de executar os passos indicados nos *Apêndices F* e *G*, o aplicativo WINOBJ deve ser utilizado para verificar se o objeto do dispositivo foi criado e está disponível. Este aplicativo é fornecido pelo SDK (*Software Development Kit*).

Para testar as rotinas de despacho, deve ser criada uma aplicação de usuário para abrir e fechar um manipulador para o driver, usando API's da biblioteca Win32. A aplicação deve solicitar serviços através dos códigos específicos de requisições, e ser executada em várias janelas do Windows NT ao mesmo tempo, para testar o gerenciamento de requisições realizadas por múltiplos chamadores.

Em várias situações, podem ocorrer erros internos que não são detectados através dos testes acima. Para tanto, o SDK fornece um utilitário chamado WINDBG (*Windows Debug)* que permite a depuração de drivers em modo kernel. Os seguintes procedimentos indicam como configurar e utilizar o WINDBG, segundo o DDK [Ddk96c] :

1) Conectar o computador origem ao destino através de um cabo serial, usando qualquer porta livre do tipo DB25 (25 pinos) ou DB9 (9 pinos), como segue :

DB 9		DB 25	
DO PINO	PARA PINO	DO PINO	PARA PINO
2	3	2	3
3	2	3	2
5	5	7	7

Figura H1 – Pinagem para ligação entre computadores usando cabo serial

2) Para se testar a funcionalidade do cabo, pode ser utilizado o aplicativo TERMINAL.EXE disponível nos Windows 95, 98 e NT.

3) No computador DESTINO :

- Instalar a versão "Testada" (*checked*) do driver;
- Ativar a depuração de *kernel* do Windows NT através da alteração do arquivo de inicialização BOOT.INI deste sistema operacional, como segue :

```
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Server Version 4.00 [Debug]" /DEBUG
/DEBUGPORT=COM2
```

4) No computador ORIGEM :

- Instalar a versão "Liberada" (*free*) do driver;
- Instalar o SDK (Software Development Kit) juntamente com o aplicativo WINDBG;
- Copiar os arquivos de símbolos do computador DESTINO para o ORIGEM, para que o utilitário WINDBG possa interpretar corretamente as informações provenientes deste primeiro computador. Para tanto se deve :
  - Criar a árvore padrão de diretórios, a partir de qualquer diretório do disco rígido :

    \SYMBOLS
    \DLL
    \EXE
    \SYS
  - Copiar arquivo RPWNT.SYS para o diretório \SYMBOLS\SYS;
  - Copiar arquivo NTOSKRNL.EXE do diretório \SUPPORT \ DEBUG \ I386 \ SYMBOLS \ EXE pertencente ao Cdrom de instalação do sistema operacional, para o diretório \SYMBOLS\EXE;
  - Copiar arquivo HAL.DBG do diretório \SUPPORT \ DEBUG \ I386 \ SYMBOLS \ DLL pertencente ao Cdrom de instalação do sistema operacional, para o diretório \SYMBOLS\DLL;
- Copiar os arquivos fontes do driver RPWNT;
- Execute o aplicativo WINDBG através de uma janela do prompt (DOS) com as seguintes opções :

start windbg -k *i386 port speed*

onde *port* é a porta serial sendo utilizada (por exemplo : COM1, COM2, ... COM*n*) e *speed* é a velocidade da comunicação a ser iniciada (por exemplo : 9600, 19200, 57600, entre outras);

- Usando a opção "*User DLLs*" do menu "*Options*", especificar o caminho de pesquisa dos arquivos de símbolos;
- Usando a opção "*Debug*" do menu "*Options*", especificar o caminho de pesquisa dos arquivos fontes do protocolo de transporte;
- Selecionar a opção "*Go*" do menu "*Run*" para aparecer a *mensagem "KD: waiting to connect..."* na janela;

5) Reinicializar o computador DESTINO e escolher a opção de depuração de kernel do Windows NT. Para que o utilitário WINDBG pare este processo (*boot*) iniciando a depuração, deve ser escolhido a opção "*Initial Breakpoint*" do menu "*Options*";

1) Para parar a depuração, tecle "CTRL+C" no computador ORIGEM.

Opções mais avançadas de depuração são explicadas em [Ddk96c].

APÊNDICE I

REPRODUÇÃO DO EXPERIMENTO

Para que o protótipo possa ser reproduzido em um ambiente Windows NT, as etapas a seguir devem ser executadas em dois computadores ligados em rede:

1. Instalação do driver de transporte (RPWNT.SYS) utilizando o arquivo auxiliar de instalação (OEMSETUP.INF), como descrito no *Apêndice G*;
2. Cópia da biblioteca auxiliadora de soquetes (WSHRPWNT.DLL) para o diretório \WINNT\SYSTEM32;
3. Reinicialização do sistema operacional Windows NT;

Em um computador (COMP_A), execute o programa simulador de ping (PINGRP.EXE) utilizando o parâmetro " *-r* " para entrar em modo de recepção de pacotes. Por exemplo:

```
pingrp -r
```

No outro computador (COMP_B), execute o programa simulador de ping (PINGRP.EXE) utilizando os parâmetros " *-d* " seguido do nome do computador destino (COMP_A), e do parâmetro opcional " *-t* " seguido do texto que poderá ser enviado. Por exemplo:

```
pingrp -d COMP_A -t "Esta mensagem e' um teste"
```

A mensagem é enviada para o computador COMP_A que mostra em tela o texto recebido e retorna pacotes indicando o recebimento com sucesso ao computador COMP_B.

O programa simulador de ping (PINGRP.EXE) apresenta uma ajuda com os parâmetros que podem ser passados para ele. Para tanto, deve ser executado sem parâmetros. Por exemplo:

```
pingrp
```