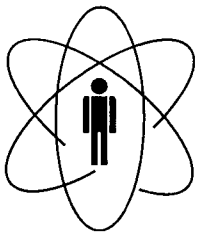
ISSN 0101-9201



CBPF - CENTRO BRASILEIRO DE PESQUISAS FÍSICAS
Rio de Janeiro

Notas Técnicas

CBPF-NT-003/96
December 1996

Acesso Remoto via PPP

Joselias Targino de Oliveira, Renato Antônio T.S. Barcelos, Marcelo Fernandes Vianna, Marcelo Portes Albuquerque, Marcio Portes de Albuquerque e Marita Maestrelli

CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico

Joselias Targino de Oliveira
joselias@cat.cbpf.br

Renato Antônio T. S. Barcelos
barcelos@cat.cbpf.br

Marcelo Fernandes Vianna
vianna@cat.cbpf.br

Marcelo Portes Albuquerque
marcelo@cat.cbpf.br

Marcio Portes Albuquerque
mpa@cat.cbpf.br

Marita Maestrelli
marita@cat.cbpf.br

PREFÁCIO

O aumento da utilização da Rede Local do CBPF, constatado nos últimos períodos, levou a CAT a repensar o modo como ela é acessada. Este aumento é limitado pelo número de estações de trabalho e/ou pela sua performance.

Máquinas de maior poder de processamento foram incorporadas à Rede aumentando, assim, a sua capacidade computacional. Porém, também seria ótimo se conseguíssemos viabilizar o acesso remoto, permitindo ao usuário executar ou acompanhar seus programas (simulações) sem a necessidade de estar presente na instituição. A apresentação do projeto de implementação do acesso remoto é o assunto desta nota técnica.

PARTE I - HISTÓRICO DO PROJETO

O sistema foi primeiramente montado em uma das estações de trabalho que compõem a Rede Sun-Unix (cat1.cat.cbpf.br - IP: 152.84.253.10), plataforma da Sun Microsystems. Naquele momento, um modem instalado na porta serial da estação permitia, através da linha telefônica, o login remoto no modo texto, emulando terminal VT100. Porém, no sentido de tentar escapar de arquiteturas proprietárias e a oportunidade de trabalhar tecnologia mais recente, remodelamos o projeto.

Foi instalado o sistema operacional Linux em uma máquina IBM-PC compatível, tendo um microprocessador I486DX-100MHz como CPU. Esse sistema operacional é uma versão Unix 32-bits para computadores baseados em microprocessadores Intel I386, I486 ou Pentium. Possui todas as ferramentas para comunicação comuns às estações de trabalho Unix (do protocolo TCP/IP ao UUCP[1]), ambiente para desenvolvimento de software (C, C++, Lisp, entre outros) e uma interface gráfica padrão X Windows baseada na filosofia WYSIWYG[2]. A vantagem de ter um sistema operacional Unix rodando em uma máquina Intel é que podemos unir recursos como multitarefa proemptiva e o multiprocessamento linear e paralelo daquele a um ambiente de arquitetura totalmente aberta, onde a busca pelo ganho de performance tem levado a microeletrônica ao estado da arte.

Para que fosse possível a oferta de melhores serviços de rede às estações remotas, viabilizamos o protocolo de comunicação PPP (Point-to-Point Protocol). Este protocolo, que faz parte do conjunto de ferramentas de comunicação do Linux, fornece mais flexibilidade do que outros protocolos de acesso. Com isso, as estações remotas conectadas à nossa Rede Local, agora através desta nova máquina (meson.cat.cbp.br - IP:152.84.253.32) podem ter o mesmo grau de interação com a Rede. Apenas é necessário disponibilizar no sistema remoto as ferramentas de comunicação apropriadas.

PARTE II - BASE TÉCNICA

2.1) Tecnologia de Internetworking

A tecnologia de **"Internetworking"** ou **"Internetting"** tem a finalidade de interconectar diversas redes físicas, com suas diferentes características, de forma transparente aos usuários e sem compromisso com um fabricante específico (arquitetura proprietária).

---

[1] **UUCP** - Unix-to-Unix Copy - Pacote de programas que basicamente copia arquivos de um host a outro sobre linha serial e também permite certas ações serem executadas em host remoto.

[2] **WYSIWYG** - What You See Is What You Get (traduz-se ao pé da letra).

2.2) Redes de Computadores

Uma **Rede** é um grupo de dois ou mais computadores ligados por um meio físico (cabos, linhas telefônicas, ondas de rádio, etc.). Quando conectados, esses computadores podem se comunicar e compartilhar seus recursos, tais quais como arquivos e impressoras.

2.3) Componentes de Rede

São basicamente três elementos:

Software de rede - Fornece a interface entre o usuário e a rede, permitindo que você utilize os recursos que estão para ser compartilhados.

Adaptador de rede - É o dispositivo que conecta fisicamente um computador à rede.

Protocolo - É a "linguagem" que os computadores usam para se comunicar na rede. São regras de procedimentos e formatos convencionados que, mediante sinais de controle, permitem o estabelecimento da comunicação.

2.4) Arquitetura TCP/IP

É o conjunto de protocolos básicos que implementa a tecnologia de internetworking. Permite o tratamento da informação transportada, independente do tipo de hardware ou dos dados roteados entre as redes, além da transparência quanto ao tipo de aplicação.

Basicamente, esse tratamento consiste na divisão de todo o processo de comunicação em etapas, onde cada uma delas é responsável por uma bem definida atividade. A isso denominamos **arquitetura em camadas**. Os dados a serem transmitidos são divididos, na camada responsável por isso, em unidades de tamanho variável denominadas **datagramas**. Informações de controle desses protocolos são adicionadas aos datagramas, em uma espécie de encapsulamento, para indicar sua origem e destino, garantir a integridade dos dados e a validação da mensagem, entre outras atividades. Feito isso, esse conjunto já está pronto para ser entregue às camadas adjacentes responsáveis pela interface do sistema com o meio físico usado para a conexão. Dali serão transmitidos na rede. No destino, a operação é simplesmente feita de modo inverso. Os datagramas são desmontados e reagrupados para que, no final do processo, tenhamos os dados originais entregue à respectiva aplicação que os solicitou (fig. 1).

2.4.1) Modelo da Arquitetura

A arquitetura TCP/IP estabelece uma estrutura em quatro camadas ou níveis conforme mostra a figura 2.

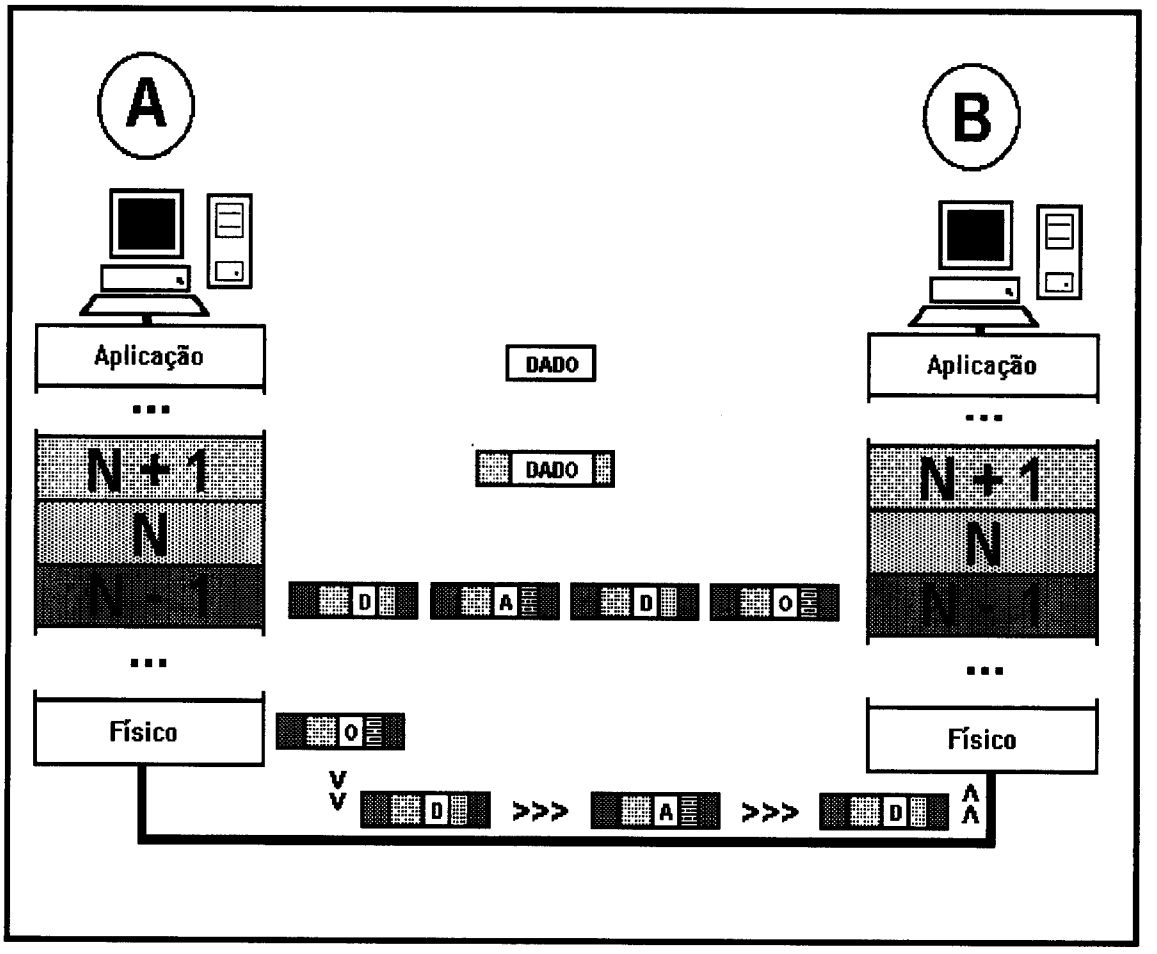


Figure 1 - Transferência de datagramas em arquitetura TCP/IP

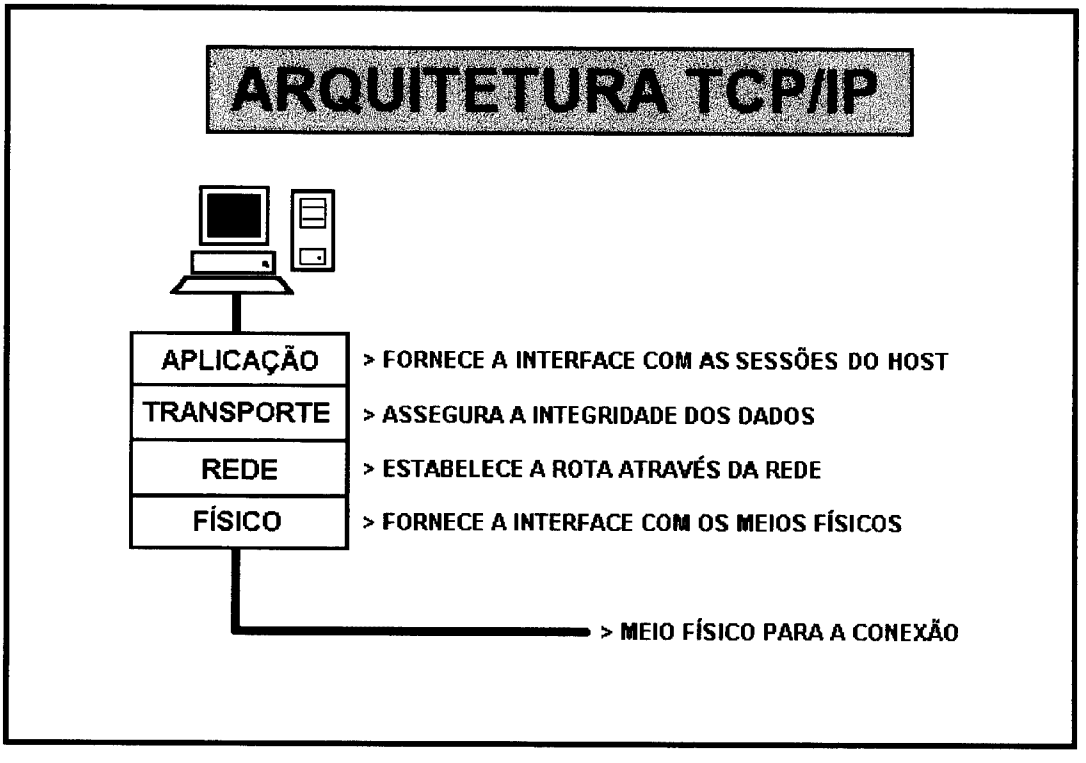


Figure 2 - Camadas da arquitetura TCP/IP

2.4.2) Camada Aplicação

O nível mais alto desta arquitetura tem a finalidade de efetuar a interface com as aplicações que desejam utilizar os meios de comunicação de dados.

2.4.3) Camada Transporte

Fornece uma forma de comunicação "fim-a-fim" entre as aplicações. Ela é responsável pela integridade dos dados transportados, implementando mecanismos de segurança, controle de fluxo, reconhecimentos, etc. É o protocolo **TCP** que implementa estas funções.

2.4.4) Camada Rede

Esta camada é responsável pela flexibilidade e poder de conectividade da arquitetura. Provê a interconexão de diversas redes, com seus diferentes meios de transmissão, topologias e protocolos de acesso ao meio. É o protocolo **IP** que implementa estas funções, além de fornecer um identificador de rede único para máquina (chamado **número IP**).

2.4.5) Camada Física

Devido ao grande número de padrões já existentes para esta camada, a arquitetura TCP/IP não padroniza um conjunto específico de protocolos de acesso ao meio e interface física, muito pelo contrário, esta arquitetura independe do meio físico utilizado. Sendo assim, podemos utilizar padrões do **IEEE 802.3, 802.4** ou **802.5**[3] para LANs, padrões como [4]**HDLC** (norma X.25 do CCITT), ou até protocolos proprietários para WANs.

---

[3] **IEEE 802** - Uma grande família de padrões referentes às conexões físicas e elétricas das redes locais, desenvolvido pelo IEEE (Institute of Eletrical and Electronics Engineers) dos Estados Unidos.

[4] **HDLC(High-Level Data Link Control)** - Um padrão amplo desenvolvido pela Internacional Standards Organization (**ISO**). O **HDLC** é um protocolo baseado em bits para a camada de link.

PARTE III - O PROTOCOLO PONTO A PONTO (PPP)

O Protocolo ponto a ponto (PPP) fornece um método padrão para transportar datagramas multi-protocolo sobre ligação ponto-a-ponto, seja ela uma linha telefônica como um Modem, ou uma linha serial direta . PPP é compreendido por três principais componentes:

1. Um método para encapsular datagramas multi-protocolo (TCP/IP, IPX, AppleTalk, etc...).
2. Possui uma série de Protocolos de Controle de Enlace (LCP) para estabelecer, configurar e testar a camada de Enlace de Dados.
3. Possui uma série de Protocolos de Controle de Rede (NCP) para estabelecer e configurar diferentes protocolos da camada de Rede.

O protocolo ponto-a-ponto, com exceção do campo de Protocolo, usa os princípios, terminologia, e estrutura de quadro do High-level Data Link Control (HDLC) da Organização Internacional de Padronização (ISO).

3.1) Operação do Link PPP

Com o objetivo de estabelecer a comunicação sobre uma ligação ponto-a-ponto, o PPP primeiro envia pacotes de Protocolo de Controle de Enlace (LCP) para configurar e testar a camada de link de dados. Após o link ter sido estabelecido, o PPP envia pacotes NCP para escolher e configurar um ou mais protocolos da Camada de Rede, como por exemplo: TCP/IP, DECnet, IPX, etc... Somente depois de cada protocolo da Camada de Rede ter sido configurado, datagramas de cada protocolo podem ser enviado sobre o link.

O link permanecerá configurado para comunicação até que um pacote LCP ou NCP feche o link, ou até algum evento externo ocorrer, e.g., tempo de inatividade expirado ou interferência do usuário.

3.2) Diagrama de Estado

No processo de configuração, manutenção e terminação; o link ponto-a-ponto, passa por vários estados distintos os quais são especificados no Diagrama de Estado simplificado na figura 3:

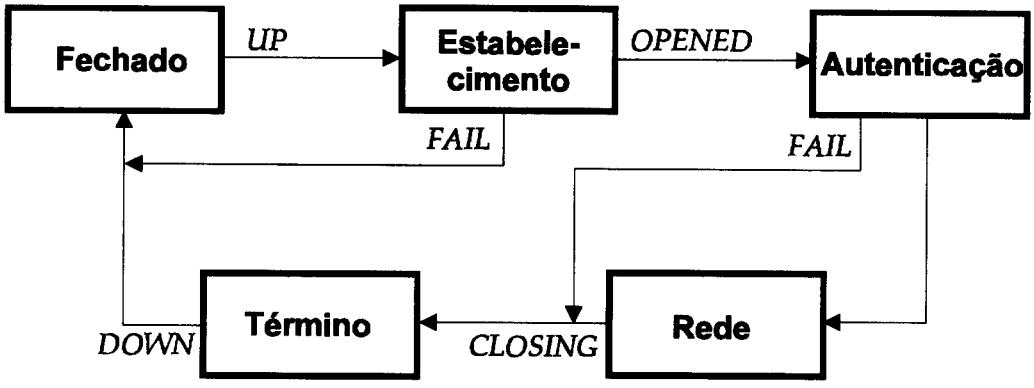


Figure 3 - Fases de Estabelecimento e Término de um Link

3.2.1) Link Fechado (Camada Física não está pronta)

O link necessariamente começa e termina nesta fase. Quando um evento externo (tal como detecção da portadora ou configuração do administrador de rede) indica que a camada física está pronta para ser usada, PPP procederá à fase de Estabelecimento de Link. A transição para a fase de Estabelecimento de Link sinalizará um **evento UP**.

Nota: O evento UP indica que a Camada Física está pronta para carregar pacotes.

3.2.2) Fase de Estabelecimento de Link

O Protocolo de Controle de Link (LCP) é usado para estabelecer a conexão através da troca de pacotes de Configuração. Ao completar esta troca entra-se no estado LCP OPENED. Todas Opções de Configuração são atribuídas pelo seus valores default, ao menos que seja alterado pela troca de configuração. Qualquer pacote recebido que não seja LCP durante esta fase deve ser discartado.

Nota: LCP OPENED é o estado da Camada de Enlace quando ela está pronta para a comunicação. Enquanto LCP estiver em estado OPENED, qualquer pacote de protocolo que não é suportado pela implementação deve ser retornado num de protocolo de rejeição (Protocol-Reject).

3.2.3) Fase de Autenticação

Em alguns links ela pode ser desabilitada, permitindo que pacotes de Protocolo da Camada de Rede sejam trocados. Por default, a autenticação não é obrigatória. O LCP também negocia a autenticação. Quando o LCP negocia autenticação de protocolos, ele determina que nível de validação de segurança o servidor de acesso remoto poderá realizar e o que o servidor exige. O PAP (Password Authentication Protocol) e o CHAP (Challenge-Handshake Authentication Protocol), são exemplos desses protocolos.

A transição da Fase de Autenticação para a Fase de Protocolo de Rede não deve ocorrer até a autenticação ser completada. Se a autenticação falhar, o

autenticador seguirá para a Fase de Terminação do link; é claro que, somente após um determinado número de tentativas ser excedido.

Somente protocolo de link, protocolo de autenticação e pacotes de monitoramento de qualidade de link são permitidos durante esta fase. Todos os outros pacotes são cuidadosamente descartados.

3.2.4) Fase de Protocolo de Camada de Rede

Uma vez que o PPP termina a fase de autenticação, cada protocolo de Camada de Rede é separadamente configurado pelo Protocolo de Controle de Rede (NCP) apropriado.

Somente após um NCP atingir o estado OPENED, PPP carregará os pacotes de protocolo de Camada de Rede correspondente.

Durante esta fase, o tráfego consiste de possível combinação de LCP, NCP e pacotes de protocolo de Camada de Rede.

3.2.5) Fase de Terminação do Link

PPP pode terminar o link a qualquer momento. Isto pode acontecer por causa da perda da portadora, falha na autenticação, falha na qualidade do link, a extinção de um tempo preestabelecido ou ação do administrador.

LCP é usado para fechar o link através da troca de pacotes de Terminação. Após a troca de pacotes de Terminação, a implementação deve sinalizar a Camada Física para desconectar. O transmissor do pacote de pedido de término (Terminate-Request) desconecta após receber um pacote de confirmação de término (Terminate-Ack). PPP segue para a fase de Link Fechado.

3.3) Formato do Protocolo Ponto a Ponto

Protocolo síncrono desenvolvido em meados dos anos 80 para atender a arquitetura **SNA** (*System Network Architecture*) em transmissões half ou full-duplex[5], com configurações ponto a ponto, em linhas comutadas ou permanentes, trabalhando com uma estrutura de quadros (frames), no formato da figura 4:

[5] **half ou full-duplex** - half-duplex é quando a comunicação é feita em ambas os sentidos, porém não simultaneamente e full-duplex a comunicação é feita em ambas os sentidos simultaneamente.

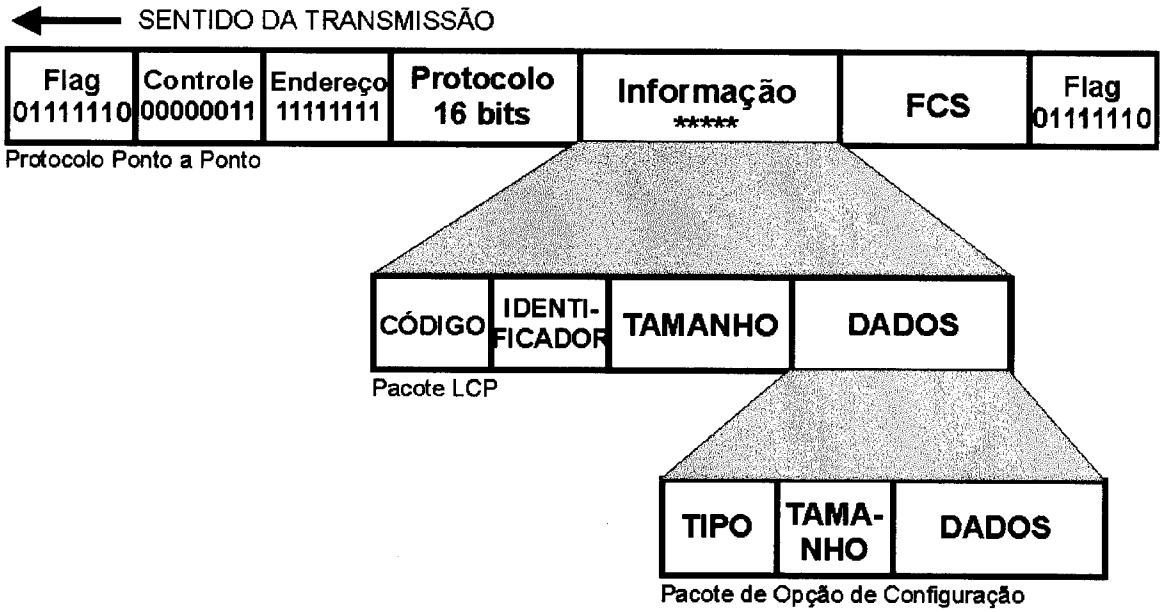


Figure 4 - PPP transportando pacote LCP

Tal Protocolo proporciona uma melhor utilização do canal de comunicação pela capacidade de operar em full-duplex, permitindo o envio de vários quadros consecutivos sem a necessidade de confirmação individual de recebimento de cada quadro por parte da estação receptora.

3.3.1) Descrição do Quadro

- Flag (ou indicador)

É um byte padrão (01111110) que delimita o início e o fim de um quadro, o que significa providenciar o sincronismo, servindo como referência para a localização dos demais campos (endereço, controle, informação, FCS). Um mesmo *Flag* pode delimitar o término e o início de outro quadro.

Nota: Quando o enlace está ativo, mas não existem quadros de dados a serem enviados, são transmitidos flags como preenchimento entre quadros.

- Endereço

O campo de Endereço é um simples byte e contém a seqüência binária 11111111 (em hexadecimal - 0xff), o endereço de "Todas-Estações". O endereço de Todas-Estações deve sempre ser reconhecido e recebido. Quadros com outros endereços devem ser descartados.

- Controle

Este campo identifica o tipo de quadro que está sendo enviado, ou seja, se é quadro de informação, de supervisão, ou não numerado. Como o Campo de Controle do PPP é um byte que contém a seqüência binária 00000011 (hex 0x03), indica que o quadro é não numerado, usado para enviar quadro de informações fora da seqüência normal de numeração.

- Protocolo

O campo de Protocolo é formado por um ou dois bytes e seu valor identifica o protocolo encapsulado no campo de Informação do pacote. O campo é transmitido e recebido primeiro pelo byte mais significativo.

Para valores na ordem de:

0*** a 3*** => Protocolo de camada de rede de pacotes específicos.
8*** a b*** => Pacotes associados à Protocolo de Controle de Rede (NCPs).
c*** a f *** => Pacotes de Protocolos de Controle de Enlace (LCP).

Valores inicialmente associados aos respectivos protocolos:

Valores (em hexadecimal)	Protocolo
0001 a 001f	Reservado (transparência ineficiente)
0021	Internet Protocol
0023	* ISO CLNP
0025	* Xerox NS IDP
0027	* DECnet Phase IV
0029	* Appletalk
002b	* Novell IPX
002d	* Van Jacobson Compressed TCP/IP 1
002f	* Van Jacobson Compressed TCP/IP 2
8021	Internet Protocol Control Protocol
8023	* ISO CLNP Control Protocol
8025	* Xerox NS IDP Control Protocol
8027	* DECnet Phase IV Control Protocol
8029	* Appletalk Control Protocol
802b	* Novell IPX Control Protocol
802d	* Reservado
802f	* Reservado
c021	Link Control Protocol
c023	Password Authentication Protocol
c025	Link Quality Report
c223	Challenge Handshake Authentication Protocol

* Reservado para o uso futuro

- Informação

O Campo de Informação contém o datagrama (informação propriamente dita) para o protocolo especificado no Campo Protocolo. A informação é de 0 ou mais bytes, com tamanho máximo determinado pelo usuário (normalmente até de 2Kbytes).

- Seqüência de Checagem de Frame (FCS)

É um teste de redundância de 16 bits usado para detecção de erros. A seqüência FCS é composta de 16 bits inseridos na transmissão, resultado das operações matemáticas entre o sinal do quadro (campo de endereço, controle e informação) e um polinômio gerador ($X^{16} + X^{12} + X^5 + 1$), segundo a técnica *Cyclic Redundancy Checking* **(CRC)**.

3.3.2) Formato do pacote LCP

Existem três classes de pacotes LCP:

1. Pacotes de configuração do link usado para estabelecer e configurar um link (Configure-Request, Configure-Ack, Configure-Nak e Configure-Reject).
2. Pacotes de terminação do link usado para terminar um link (Terminate-Request e Terminate-Ack).
3. Pacotes de Manutenção do link usado para gerenciar e recuperar um link (Code-Reject, Protocol-Reject, Echo-Request, Echo-Reply, e Discard-Request).

Um pacote LCP é encapsulado no campo de informação, onde o campo de protocolo PPP indica o tipo hexadecimal 0xC021 (protocolo de controle de link).

Um resumo do formato do pacote LCP é mostrado abaixo. Os campos são transmitidos da esquerda para direita.

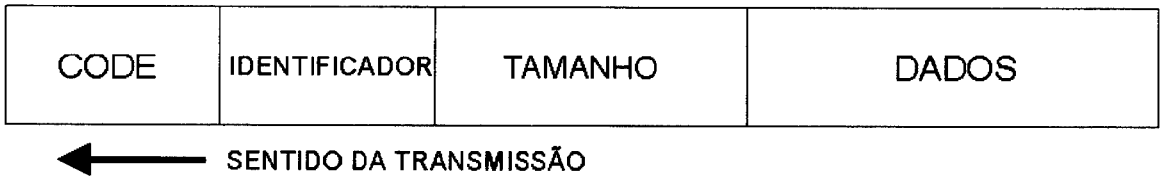


Figure 5 - Pacote LCP

- Código

Campo Código é um byte, e identifica o tipo de pacote LCP. Quando um pacote é recebido com um campo de código desconhecido, um pacote de Code-Reject é transmitido. Os valores com seus respectivos pacotes LCP estão enumerados abaixo:

Valores Pacotes LCP

1	Configure-Request
2	Configure-Ack
3	Configure-Nak
4	Configure-Reject
5	Terminate-Reject
6	Terminate-Request
7	Code-Reject
8	Protocol-Reject
9	Echo-Request
10	Echo-Reply
11	Discard-Request

- Identificador

O campo identificador é um byte, e auxilia a combinar pedidos e repetições. Quando um pacote é recebido com um campo identificador inválido, o pacote é cuidadosamente descartado.

- Tamanho

O campo Tamanho é composto por dois bytes, e indica o tamanho de pacote LCP, incluindo os campos Código, Identificador, Tamanho e Dados. O tamanho não deve exceder a Unidade de Recepção Máxima (MRU) do link.

Bytes fora da faixa do campo Tamanho são tratados como excesso e são ignorados na recepção. Quando um pacote é recebido com um campo Tamanho Inválido, o pacote é descartado.

- Dados

O campo Dados é composto por zero ou mais bytes, como indicado pelo campo Tamanho. O formato do campo Dados é determinado pelo campo Código. Neste campo estão as Opções de Configuração do LCP, conforme veremos depois.

3.3.3) Transferência de Informações LCP

Após os MODEMS completarem uma ligação e o usário entrar com uma conta PPP, o agente do protocolo (PPPd, que será mencionado na parte IV desta Nota), em ambas as máquinas, tentará iniciar uma conexão, enviando pacotes LCP. Para que as mesmas entrem em acordo é importante que uma delas, e somente uma, aguarde passivamente um pacote LCP válido, para somente então responder com outro pacote LCP.

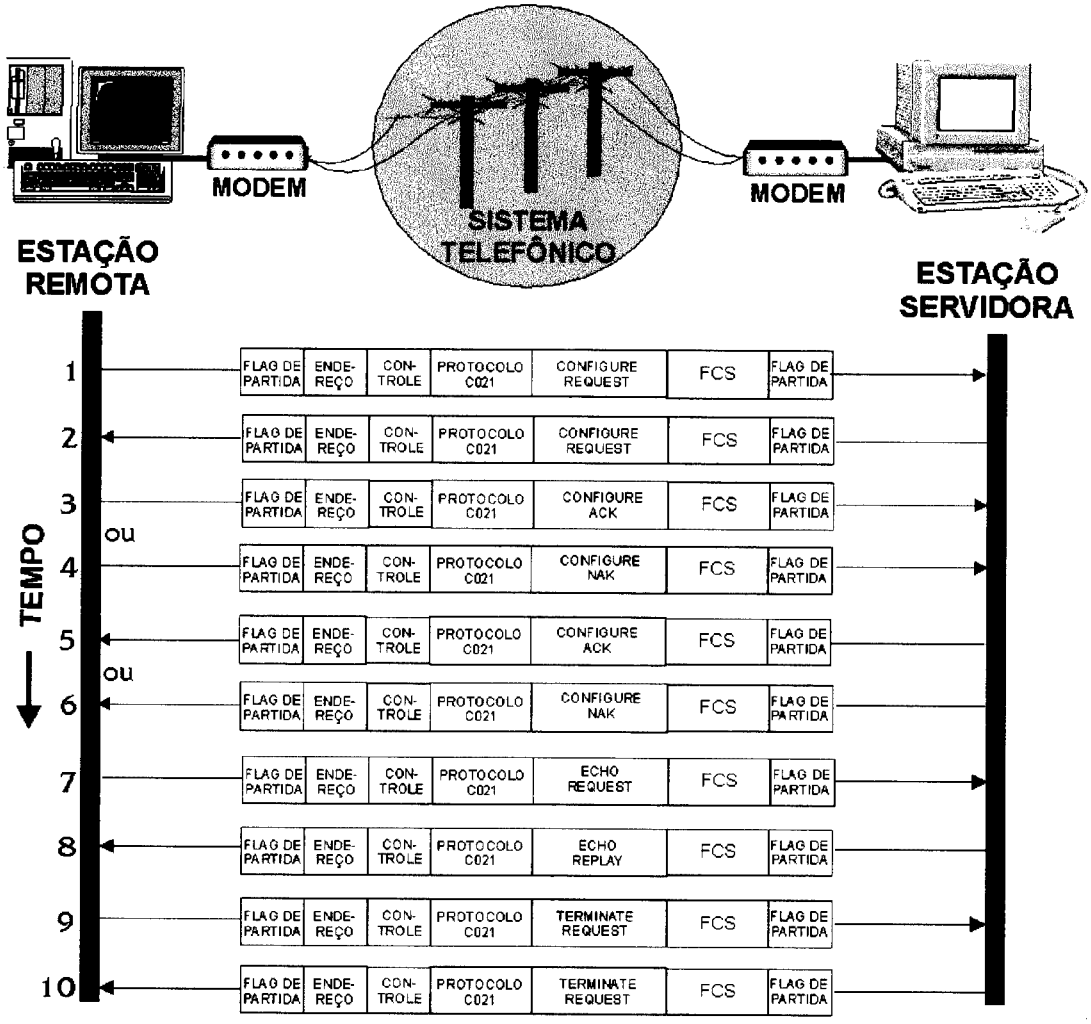


Figure 6 - Transferência de pacotes LCP em função do tempo

1. A Estação Remota envia um pacote *Configure-Request* para abrir uma conexão com a colocação especificada em Opções de Configuração. O temporizador é iniciado, para reenviar pacotes de *Configure-Request* caso o primeiro se perca.

2. A Estação Servidora recebe o pacote e envia outro indicando se as Opções de Configuração do *Configure-Request* foi aceitável ou não.

3. A Estação Remota envia um pacote de reconhecimento da recepção de pacotes *Configure-Request* com a configuração aceitável.

4. Caso a configuração não seja aceitável, a Estação Remota envia um pacote *Configure-Nak* ou *Configure-Reject* para refutar um valor da Opção de Configuração e sugerir um novo, com valores aceitáveis.

5. A Estação Servidora recebe um pacote *Configure-Ack* válido e reenvia-o como uma resposta positiva ao pacote *Configure-Request.* Após esta fase as Estações estão prontas para trocar pacotes NCPs.

6. A Estação Servidora recebe um pacote *Configure-Nak* ou *Configure-Reject* e reenvia-o como uma resposta negativa ao pacote *Configure-Request.*

7. Mesmo durante a Fase de Protocolo da Camada de Rede, pacotes LCPs do tipo *Echo-Request* ou *Discard-Request* podem ser transmitidos. Estes pacotes são úteis como um auxílio em reconstrução, determinação da qualidade do link, teste de desempenho, e uma série de outras funções.

8. A Estação Servidora reconhece a recepção de um pacote *Echo-Replay.*

9. A Estação Remota envia um pacote *Terminate-Request,* indicando o desejo de fechar a conexão.

10. A Estação Servidora recebe o pacote de término de conexão e reenvia-o indicando que um pacote *Terminate-Request* foi recebido e fechando o link.

3.3.4) Opções de Configuração do LCP

Opções de Configuração do LCP contém informações necessárias para negociar as características do link e testá-lo. As duas opções mais importantes que podem ser negociadas pelo LCP são a Unidade de Recepção Máxima (MRU) e o Mapeamento de Controle de Caracter Assíncrono. Se uma Opção de Configuração não está incluída num pacote *Configure-Request,* o valor default para Opção de Configuração é assumida.

O fim da lista de Opções de Configuração é indicada pelo campo Tamanho do pacote LCP. Um resumo do formato de Opção de Configuração é mostrado abaixo.

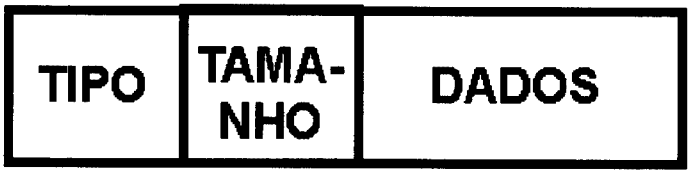


figura 7 - Pacote de Opções de Configuração do LCP

- Tipo

O campo Tipo é um byte, e indica o tipo de Opção de Configuração.

0	RESERVADO
1	Maximum-Receive-Protocol
3	Authentication-Protocol
4	Quality-Protocol
5	Magic-Number
7	Protocol-Field Compression
8	Address-and-Control-Field-Compression

- Tamanho

O Campo Tamanho é um byte, e indica o tamanho desta Opção de Configuração incluindo os campos: Tipo, Tamanho e o Dado.

Se uma Opção de Configuração negociável é recebida num *Configure-Request*, porém com um Tamanho inválido ou não reconhecido, um pacote *Configure-Nak* deve ser transmitido incluindo a Opção de Configuração desejada com um Tamanho e Dados apropriado.

- Dados

O campo Dados é composto de zero ou mais bytes, e contém a informação específica para a Opção de Configuração. O formato e tamanho do campo Dados é determinado pelo campo Tipo e Tamanho.

3.3.5) Protocolo de Controle de Rede (NCP) PPP

Os Protocolos de Controle de Rede estabelecem e configuram diferentes parâmetros de protocolo de rede. O tipo de Protocolo de Controle de Rede que o PPP seleciona depende de que protocolo (NetBEUI, TCP/IP, IPX, etc...) está sendo utilizado para estabelecer a conexão, e.g., O NetBIOS Frames Control Protocol (NBF CP) é utilizado para configurar, ativar e desativar o protocolo NetBEUI em ambas as extremidades do enlace. O Internet Packet Exchange Protocol (IPXPC) é para

configurar, ativar e desativar módulos de protocolo IPX, e o Internet Protocol Control Protocol (IPCP) é para configurar, ativar e desativar módulos de protocolo IP.

Para que datagramas IP sejam enviados através do link, as máquinas que estão se comunicando com o PPP devem primeiro negociar qual endereço IP cada um delas usará. O protocolo de controle para isto é IPCP, o Protocolo de Controle de Protocolo Internet.

IPCP usa o mesmo mecanismo do Link Control Protocol (LCP), com as seguintes exceções:

- **Campo de Protocolo da Camada de Link de Dados** - Quando um pacote IPCP é encapsulado no Campo de Informação o Campo de Protocolo indica o tipo hex 8021 (IP Control Protocol).

- **Campo Código** - Somente códigos 1 até 7 (*Configure-Request*, *Configure-Ack*, *Configure-Nak*, *Configure-Reject*, *Terminate-Request*, *Terminate-Ack* e *Code-Reject*) são usados. Outros códigos devem ser tratados como irreconhecíveis e resultar em *Code-Rejects*.

- **Tipos de Opções de Configuração** - Possui o mesmo formato do LCP porém com colocações diferentes, as quais são sitadas abaixo:

  1 IP-Addresses
  2 IP-Compression-Protocol

  O IP-Adresses da Opção de Configuração fornece um caminho para negociar os endereços IP a serem usados em cada fim do Link. Um endereço enviado como zero é interpretado como o servidor fornecê-lo. O IP-Compression-Protocol fornece uma caminho para negociar o uso de um protocolo de compressão, e.g., ***o Van Jacobson Compressed TCP/IP***[6]. Por default, a compressão não é habilitada.

[6] Van Jacobson Compressed TCP/IP reduz o tamanho dos cabeçalhos TCP/IP em pequenos 3 bytes.

PARTE IV - IMPLEMENTAÇÃO DO PROJETO

As próximas seções deste documento assumirão, em alguns momentos, que o leitor possui algum conhecimento do sistema UNIX e, em raros momentos, de programação nesse sistema. Os conhecimentos de programação, no entanto, não são necessários e apenas facilitam a compreensão de alguns comentários.

4.1) Uma conexão PPP.

Ao contrário do que possa parecer, um "link" PPP não necessariamente ocorre sobre ligações telefônicas e nem mesmo requer "contas" de usuários. É muito comum alguns acreditarem, que no momento em que estabelecem contato com seu provedor de acesso, que estão "logados" no mesmo sentido conotativo. No sentido denotativo, é evidente que o provedor tem um "logbook" das ligações para fins de controle, no entanto, o usuário não possui nenhum processo em execução no servidor de seu provedor, a excessão do processo responsável pela manutenção do "link" (pppd). Em termos claros e filosoficamente falando, uma conexão do tipo PPP simplesmente emula um cabo ethernet ou outro tipo de rede sobre a linha telefônica tornando a máquina realizadora da chamada "visível" na rede do provedor.

Uma conexão SLIP ou PPP pode ocorrer em outros meios que não por via telefônica, como por exemplo sobre uma linha serial direta entre duas estações ou sobre um "link" de rádio ou sobre qualquer outro meio imaginável de comunicação de computadores. Mesmo que por contraditório ou sem fins práticos que possa parecer, um "link" deste tipo pode ser feito sobre uma ligação real de rede ou sobre uma pseudo-ligação de um sistema para ele mesmo (net-loopback). Ligações PPP podem ter um caráter permanente ou temporário, ligações telefônicas obviamente se classificam no segundo caso.

Esta flexibilidade torna possível, por exemplo, uma empresa cuja sede se encontra em certo Estado **permanentemente** ligada em rede com sua filial, por meio de uma linha privada (dedicada).

Sistemas UNIX são extremamente flexíveis nesse sentido, cada um dos mencionados meios de ligação entre estações aparece em cada sistema sob a forma de devices de caracteres (tty-devices: /dev/tty???, /dev/cua???, ...). Assim o processo responsável pela manutenção do "link" (O ppp daemon que será mais detalhadamente descrito em uma seção posterior deste documento) nada tem a ver com o meio da transmissão, dado que todos aparecem para ele sob a mesma forma. Outros sistemas, como o MSDOS, tratam os devices de forma menos padronizada, assim os processos responsáveis` pela manutenção do link são menos genéricos, normalmente interagindo sobre os mesmos através de chamadas a BIOS ou diretamente sobre o Hardware, limitando a gama de possíveis meios a um grupo de periféricos, por exemplo, portas seriais.

Em ligações de caráter permanente, a implementeção torna-se bastante simples: Os administradores das estações que servirão como "gateways" do "link" iniciam em ambas um processo (que como citado anteriormente será descrito em uma seção posterior deste documento) responsável pelo encapsulamento dos datagramas de rede; e o "link" está estabelecido. Neste caso não faz sentido classificar uma estação como servidora e outra como cliente, as duas estão simplesmente ligadas entre si. Tal procedimento pode ser automatizado incluindo nos scripts de inicialização de cada estação a chamada ao mensionado processo.

Em ligações de caráter temporário, mais especificamente, ligações telefônicas, como as de um serviço de provedoria, o administrador da estação servidora (agora sim, a estação que disponibiliza o serviço para uma infinidade de outras estações) não está presente a cada ligação, e portanto o procedimento precisa ser automatizado. Por questões de segurança é ainda recomendável que o procedimento seja individualizado (as mencionadas "contas de usuario" para o serviço). Esta automatização é dada por meio da criação de uma ou mais contas com o propósito único de iniciar o processo de controle do PPP, assim este tipo de conta não disponibiliza ao usuario nenhum "shell", pelo contrário, esta conta não existe com o proposito de interagir com o usuário, e sim com um processo equivalente que reside na estação que realizou a chamada.

Três abordagens distintas podem ser adotadas na implementação de um serviço do tipo provedoria:

CONTA GENÉRICA - A mais simples e certamente a menos recomendada. Como o próprio nome sugere esta abordagem é não individualizada. Neste tipo de implementeção, é criada uma conta única de usuário para o serviço, evidentemente com uma única senha. Como um serviço deste tipo visa atender a determinado grupo de pessoas e não a toda a humanidade, este tipo de conta consiste em uma grave fraqueza de segurança, além, evidentemente, de inviabilizar qualquer tipo de controle sobre as responsabilidades de danos ou mal-usos do serviço. O único objetivo deste tipo de implementação se dá na fase de testes e ajustes do serviço.

CONTA INDIVIDUAL - Equivalente em funcionalidade a CONTA GENÉRICA, porém para cada um dos usuários se associa uma conta com senha particular, dessa forma o usuário é inteiramente responsável pela segurança de sua senha e, ainda, o administrador do serviço é capaz de controlar o bom ou mal uso do serviço consultado os "logs" do sistema. Esta forma de implementeção permite o uso dos procedimentos de autenticação PAP/CHAP inerentes ao protocolo PPP e viabiliza outros procedimentos avançados como por exemplo um serviço de call-back. Sem dúvida esta é a forma mais comumente utilizada em serviços de provedoria e o grau de segurança varia de bom à excelente quando utilizada autenticação e/ou call-back.

SCRIPT DE ATIVAÇÃO - Este metodo é bem menos comum e seu uso atende normalmente a serviços de provedoria de pequeno porte como a relação empresa-funcionário e implica na já existência de contas comuns de usuários (ou conta shell) do sistema. Consiste em permitir que o usuário realize uma conexão do tipo terminal com a estação servidora e, em um momento a seu dispor, inicie sobre a mesma um "link" do tipo PPP através de uma chamada a um script que realiza isso automaticamente. As

abordagens do tipo ***CONTA GENÉRICA*** e ***CONTA INDIVIDUAL***, em verdade, realizam uma chamada a um script desse tipo, sem no entanto passar por um shell. A segurança desse método é quase a mesma de uma conta shell, exceto pelo fato de possuirem os usuários contas normais no sistema. A segurança é, portanto, inversamente proporcional ao número de usuários. Não faz sentido a utilização de autenticações nesse caso uma vez que o usuário já está dentro do sistema. O uso de call-backs, neste modelo, pode ser visto simplesmente como uma conveniência e não como um método de segurança.

4.2) Considerações sobre endereços de rede (IP address ou endereço Internet)

Está implícito que todas as máquinas de uma rede possuem um endereço cuja relação é biunívoca (uma máquina é representada por um só endereço, e um endereço representa uma só máquina). Uma conexão PPP, seja ela permanente ou temporária, não pode em momento algum violar esta lei de conservação de endereços.

Em "links" estáticos não há muito o que se ponderar a esse respeito, porém em "links" dinâmicos, como em serviços de provedoria em que vários usuários podem utilizar o serviço ao mesmo tempo, isso representa um sério problema dado que podem surgir conflitos na atribuição de IPs.

Como foi explicado na descrição do Protocolo Ponto a Ponto, as entidades que controlam o fluir da comunicação estabelecem na fase inicial da conexão uma negociação na qual decidem os endereços pelos quais uma estação reconhecerá a outra (dito desta forma porque em casos especiais, embora não seja conveniente adotar tal postura, tais endereços não necessariamente são os endereços reais, i.e., os reconhecidos pelas demais estações da rede). Em um serviço do tipo provedoria é sumariamente recomendado que somente a estação chamada de servidora controle os endereços, tomando a estação cliente uma postura passiva, acolhendo os endereços ditados pelo provedor, evitando conflitos no momento da conexão que levarão ao não comum acordo e corte da mesma). Assim podemos dizer que a estação servidora ATRIBUI à cliente um endereço.

Uma vez escolhido um grupo de endereços a serem dedicados ao serviço, e supondo que existe mais de uma linha física ligada à estação servidora, é necessario evitar que jamais sejam atribuidas a duas linhas (maquinas remotas) distintas o mesmo endereço.

Caracterizado desta forma, tornar-se-á claro que existem duas formas distintas de realizar tal atribuição de endereços: dinâmica e estática (não confundir com ligação permanente ou temporária). Um sistema pode ainda adotar uma forma mista.

A **atribuição estática** de IPs consiste em associar a cada usuário um endereço IP, assim, partindo do princípio que nenhum usuário conectar-se-á ao provedor por

duas linhas distintas simultaneamente[7] nenhum conflito ocorrerá. O custo disso é, no entanto, consideravelmente alto dado que o número de endereços dedicados ao serviço pode vir a ser muito grande, em uma rede classe C ou uma sub-rede de uma rede classe B costumam existir apenas 256 endereços, subtraindo-se o da própria estação servidora e o do roteador, mesmo que todo um braço de 256 endereços de uma rede classe B fosse dedicado ao serviço, ainda assim o número de usuários estaria limitado a 254, e supondo que um serviço para atender a esse número de usuários deva possuir aprox. 10 linhas para atender a demanda, menos de 5% desses endereços estarão em uso a cada instante, o que definitivamente pode e deve ser considerado um desperdício. Existem porém outros benefícios associados a atribuição estática de endereços, são alguns deles: a possibilidade da inclusão de uma máquina remota no DNS ou servidor de nomes, a continuidade de uma transmissão de arquivo por FTP mesmo após a quebra e o reestabelecimento da ligação, etc..

A **atribuição dinâmica** de IPs consiste na dissociação dos endereços IP dos usuários, e pode ser implementada de algumas formas diferentes: através de um POOL de IPs que são alocados e desalocados "on-demand", que além de difícil implementeção pode criar problemas de "dead-address", quando após o fechamento de uma conexão o seu endereço é incorretamente desalocado, ou através da associação de um endereço IP para cada linha física do servidor. A atribuição dinâmica de IPs implica na não garantia de um usuário receber o mesmo endereço em duas ligações diferentes. É oportuno mencionar que mesmo utilizando-se atribuição dinâmica, ainda assim, é conveniente atribuir-se nomes aos endereços no DNS, dado que alguns serviços Internet dependem disso. A diferença que existe nesse caso em relação a atribuição estática, consiste no fato de que a máquina de um usuário não terá sempre o mesmo nome a cada conexão e, não obstante, o nome atribuido "on-the-fly" certamente não será o mesmo que o nome real da máquina, no caso de esta estar rodando um sistema UNIX, o que pode, para alguns serviços, confundi-lo. É então recomandado que, na medida do possível, aos usuários que utilizam UNIX em suas máquinas remotas sejam atribuidos IPs estáticos, mesmo em um sistema cuja regra geral seja atribuição dinâmica (o mencionado sistema misto).

4.3) Configuração do modem e do terminal.

Uma das tarefas mais trabalhosas da implementeção de um serviço como o descrito neste documento é sem dúvida a configuração do modem. Esta seção, no entanto tornará esta tarefa consideravelmente simples.

O primeiro grande problema com relação a configuração do modem é a não padronização. Essencialmente, todos os modems encontrados no mercado são o que se

[7] Existindo porém uma excessão: EQL (serial line balanced connection) em que duas linhas são utilizadas em uma única conexão PPP com o propósito de aumentar a velocidade da conexão, note porém, que para ambos os sistemas o conjunto físico das duas ligações é entendido como uma única ligação de maior velocidade. Este documento no entanto não entrará em maiores detalhes quanto a esta tecnologia.

chama de "Hayes compatible", porém todas as características que foram surgindo com o tempo, após esta "padronização" do *Hayes* são configuradas de forma diferente em modems de marcas distintas, i.e., um modem USRobotics spotster 14400 e um Zoltrix 14400 são ambos compatives com o *Hayes*, mas não são compatíveis entre si.

Após muito tempo configurando e otimizando a configuração do modem utilizado na implementeção do serviço de acesso remoto do CBPF (Um USRobotics Sportster 28800 bps externo) conseguimos minimizar ao máximo as strings de configuração.

Para a viabilização do serviço aqui documentado, é necessário que a estação servidora seja capaz de receber ligações na forma de terminal. E nesse contexto que se dá toda a configuração do modem.

Para facilitar a compreensão, vamos descrever brevemente como ocorre o processo de login em um terminal UNIX (note que o próprio console é considerado um terminal composto por seu teclado e monitor de vídeo).

O **init**, pai de todos os processos, é responsável pelo lançamento de todos os processos automáticos do sistema, i.e., todos os 'daemons' de rede, scripts de boot, etc. É também responsável pelo controle dos mesmos no que diz respeito ao estado da máquina (Runlevels nos SysV compliants e Single/Multiuser nos BSD compliants). O arquivo de configuração do **init** reside no diretório "/etc" com o nome de "inittab".

O Linux é uma espécie de fusão dos modelos BSD e SysV, incorporando as melhores características de cada um. Nele o init pode ser configurado para seguir tanto um modelo quanto outro. Na prática, a regra é utilizar o estilo SysV de controle de processos (O chamado SysV-like init), dado que é o mais comumente encontrado nas distribuições deste sistema operacional e também o mais poderoso.

Nos convém mostrar neste documento somente a parte do "inittab" correspondente ao controle de terminais. Abaixo, segue um exemplo do fragmento deste arquivo:

---------- Corte aqui ----------
c1:12345:respawn:/sbin/agetty 38400 tty1 linux
c2:234:respawn:/sbin/agetty 38400 tty2 linux
c3:234:respawn:/sbin/agetty 38400 tty3 linux
c4:234:respawn:/sbin/agetty 38400 tty4 linux
c5:234:respawn:/sbin/agetty 38400 tty5 linux
c6:234:respawn:/sbin/agetty 38400 tty6 linux
S0:234:respawn:/sbin/getty ttyS0 38400 vt100
---------- Corte aqui ----------

Linhas iniciadas com o caractere "#" são consideradas pelo **init** como comentários. Os campos de cada linha são separados pelo caractere "**:**".

O primeiro campo corresponde a identificação do processo. A identificação de um processo jamais deve se repetir em outro. O número de caracteres da identificação

varia de sistema para sistema, no nosso caso, o limite é de 2. Para alguns processos, em especial no "getty" mas não no "agetty" esta identificação deve seguir uma regra rígida: a identificação deve corresponder ao sufixo do tty-device utilizado, e.g., para /dev/ttyS1 a identificação deve ser S1 (Na prática, comprovamos que esta regra não precisa ser seguida).

O segundo campo corresponde aos "runlevels" (estados de execução do sistema). Cada processo é iniciado ou terminado de acordo com os runlevels listados nesse campo e com runlevel atual.

O terceiro campo corresponde a ação que deve ser tomada com relação ao processo:

"once" - O **init** lança o processo uma única vez a cada mudança para um runlevel listado no segundo campo;

"respawn" - Semelhante ao "once", exceto que o init relança o processo toda vez que o mesmo termina e não há uma mudanca de runlevel;

"wait" - Semelhante ao "once", exceto que o init espera o fim do processo antes de prosseguir com o lançamento de outros processos correspondentes ao runlevel corrente;

"boot" - O processo é iniciado toda vez que o sistema estiver para ser re-inicializado. O segundo campo é ignorado;

"bootwait" - Como o nome sugere, é uma fusão do "boot" com o "wait";

"off" - O **init** nunca lança o processo (equivale a comentar a linha, exceto pelo fato de definir uma identificação);

Existem outros possíveis valores para o terceiro campo, porém não é o objetivo deste documento ir a fundo na configuração do init. Em verdade a única ação que nos interessa é a "respawn".

Finalmente, o quarto campo é a linha de execução do processo.

Existe um grupo de processos que podem ser utilizados para iniciar um login, são alguns deles: getty, uugetty e agetty. Cada qual com suas características e formas de configuração próprias. O agetty possui uma série de vantagens para tratar terminais permanentemente ligados a estação, mas é falho no que se refere a terminais temporários, como é o caso de um modem. Para terminais temporários nos restam duas opções, o getty e o uugetty. Os dois são muito semelhantes e a única diferença consiste no fato de que o segundo utiliza "lock-files" para controlar o uso dos tty-devices. Para evitar problemas com terminais sem uso sendo incorretamente "trancados" recomendamos o uso do getty.

O objetivo de processos como o getty é iniciar uma seção de login de usuário. Caracteristicas comuns a esse tipo de processo são mostrar no terminal o arquivo

"/etc/issue" e apresentar um prompt requisitando o login (nome) do usuário, passando-o adiante ao processo "login", que por sua vez espera pela senha do usuário e, em caso de sucesso, mostra o conteúdo do arquivo "/etc/motd" e inicia um shell.

Tanto a forma como o "(??)getty" inicia o processo "login" quanto a que o "login" inicia o shell é através de uma das funções do grupo "execv", que substitui o processo corrente pelo chamado, mantendo, porém, o mesmo PID (process identification). Isto significa que todos os três (getty, login e shell) possuem o mesmo número, assim o init não considera a transformação do getty em login e desse em shell como um fim de processo, não relançando o getty como seria esperado em consequência da ação "respawn", até que o shell termine.

Nossos esforços se concentram agora na configuração do processo getty. Por ser o primeiro a interagir com o tty-device, é este que precisa definir corretamente os parâmetros de comunicação, como: velocidade de transferência, largura das palavras (leia-se número de bits por unidade, e.g., uma palavra de 8 = 1 byte), modo de paridade, etc.

4.3.1) Configuração do getty.

A linha de comando do getty segue a forma:

getty **<tty> <descricao> <terminal>** , onde

<tty> É o nome do tty-device a ser utilizado (no caso de modems, valores comuns são: ttyS0, ..., ttyS3, cua0, ..., cua3; dependendo da porta serial na qual se encontra o modem;

<descrição> É o índice de uma tabela (definida em um dos seus arquivos de configuração) que descreve os parâmetros da linha serial utilizada;

<terminal> É o nome da definição do terminal utilizado (precisa estar listado no arquivo "/etc/termcap" nos BSD ou "/etc/terminfo" nos SysV. Mais uma vez, o Linux pode estar configurado de qualquer uma das duas formas, sendo a mais comum a utilização do termcap. Para modems sugerimos o VT100 como o protocolo do terminal.

O arquivo que contém a tabela de descrições fica no diretorio "/etc" com o nome "gettydefs". Nele podem existir várias descrições, cada qual com seus parâmetros (bits de paridade, bits de parada, comprimento da palavra, etc.). Recomendados a leitura da página de manual on-line do gettydefs para entender os detalhes da configuração deste arquivo. Abaixo segue uma cópia do arquivo que nós utilizamos no CBPF:

---------- CORTE AQUI ----------
vc# B9600 SANE # B9600 SANE -ISTRIP CLOCAL #@S login: #vc
38400# B38400 CS8 CRTSCTS # B38400 SANE -ISTRIP CRTSCTS #@S login: #38400
19200# B19200 CS8 CRTSCTS # B19200 SANE -ISTRIP CRTSCTS #@S login: #19200

9600# B9600 CS8 # B9600 SANE -ISTRIP #@S login: #9600
2400# B2400 CS8 # B2400 SANE -ISTRIP #login: #2400
1200# B1200 CS8 # B1200 SANE -ISTRIP #login: #1200
300# B300 CS8 # B300 SANE -ISTRIP #login: #300
---------- CORTE AQUI ----------

Cada linha corresponde a uma descrição. São os valores do primeiro campo que são utilizados no segundo argumento do getty. Note que o getty tenta, no caso de um modem, determinar a velocidade real na qual se deu a conexão, podendo utilizar outra descrição que não a passada como argumento. Nesse contexto, a descrição passada na linha de comando pode ser entendida como a opção default. Quando a conexão é efetivada, o getty captura a mensagem do modem separando a string correspondente a velocidade. Caso o argumento de linha não case com o valor detectado do modem, o getty continua a leitura deste arquivo até encontrar a descrição apropriada, por isso este arquivo é organizado na ordem decrescente.

Normalmente o modem ao completar o "handshaking" devolve uma string semelhante a "CONNECT 14400/ARQ/MNP5/V32BIS". Aqui começam nossos problemas. O metodo utilizado pelo getty para isolar o valor da velocidade é esperar pela palavra CONNECT, considerando todo o resto até o fim da linha como descrição. Precisamos portanto configurar o modem para enviar simplesmente algo como "CONNECT 14400" de modo a não confundir o getty.

Nota1: Como o leitor deve ter notado, no arquivo de descrições não existe uma linha para a velocidade 14400. Velocidades como 14400 e 28800 são consideradas não padrão, e a regra de sua utilização é utilizar o valor padrão imediatamente acima.

Nota2: Em verdade, como veremos pouco adiante, o metodo de separação da velocidade e inteiramente configurável, porém é imensamente mais simples fazê-lo desta forma.

A partir deste ponto, devido ao já mensionado problema de compatibilidade entre modems, vamos partir do princípio que o modem utilizado e um USRobotics sportster 28800 externo (pelo simples motivo de ter sido este modem o que nós utilizamos).

Existem diversas formas de se configurar o modem no contexto de fazê-lo por intermédio do getty. Os dois extremos dessas formas são: Configurar o modem individualmente através de comandos e de suas microchaves (interruptores presentes nos modems externos) e gravar tal configuração em sua memória não volátil (NVRAM), restando para o getty a simples tarefa de resetar o modem a cada sessão (enviando a string "ATZ"); ou configurar o getty para enviar todas as strings de configuração toda vez que for iniciada uma sessão. Outras formas **não** recomendadas são os pontos intermediários, onde parte é gravada na NVRAM, parte é inicializada por sessão. A opção por uma das extremidades deve levar em consideração o seguinte critério de avaliação: Se o modem tem o propósito exclusivo de atender a um serviço do tipo provedoria, deve-se configurá-lo utilizando a sua NVRAM, senão deve-se fazê-lo da outra forma. Optamos pela primeira.

Suponhamos, agora, que o modem já está configurado. Nos resta ainda indicar ao getty o procedimento a ser adotado no princípio e no fim de cada conexão. Os arquivos de script do getty (que nada tem a ver com scripts de shell) residem no diretório "/etc/defaults". Estes arquivos podem ter os seguintes nomes: getty ou getty.<ttyname>. O arquivo sem extensão é utilizado sempre que o getty utiliza um tty-device que não possui um getty.<ttyname>, já o segundo caso se refere a configuração individualizada de cada tty-device. Se o sistema utiliza getty tanto para modems como para terminais permanentes, recomendamos EXPRESSAMETE que NÃO se altere o arquivo "getty", pois o procedimento de inicialização de um console virtual é absolutamente diferente do de um modem. A regra geral é configurar individualmente. Suponhamos ainda que o modem está ligado na primeira porta serial: ttyS0 (ou COM1 no DOS). O nosso arquivo de configuração por conseqüência é o "/etc/defaults/getty.ttyS0"; chamamos este arquivo de arquivo de configuração da linha.

O arquivo de configuração da linha pode possuir uma infinidade de opções, recomendamos, a leitura da página de manual on-line "man" do getty para informações não descritas nesse documento. Segue-se a descrição de algumas de suas opções:

CLEAR=valor - Especifica se o getty deve ou não apagar a tela antes de mostrar o conteúdo do arquivo "/etc/issue". Possíveis valores são "NO" e "YES". Se o terminal utilizado possuir a capacidade de apagar a tela (como o VT100 no nosso caso), pode-se utlizar esta opção, do contrário, como o terminal "dumb", não.

HANGUP=valor - Especifica se o getty deve ou não forçar o corte de uma ligação previamente existente no momento de seu lançamento, isto é, força o modem a desligar para que o getty tenha total controle da situação e aguarde uma ligação. Possíveis valores: "NO" e "YES".

INIT=string - Especifica uma seqüência do tipo "espera por", "envia" separada por espaços. Isto é, se a seqüência for "A" "B" "C" "D", o getty espera por "A" e então envia "B", espera por "C" e então envia "D". Como a nossa opção de configuração foi baseada na memória não volátil esta opção fica bem simples comparada à escolha de configurar o modem pela mesma.

WAITFOR=string - Faz com que o getty aguarde uma string antes de prosseguir com a execução do script.

CONNECT=string - Especifica a máscara (ou filtro) para o getty separar o campo da velocidade da mensagem do modem. É recomendada a leitura da página de manual on-line do getty para maiores informações. Como a opção INIT <string>, também é uma seqüência do tipo "espera por" "envia". São alguns dos caracteres de controle utilizados nesta opção:

\s - represnta um espaço;
\n - representa uma mudança de linha;
\r - representa o caractere <RETURN>
\A - representa o campo da velocidade.

A string default desta opção é "CONNECT\s\A\r\n".

O arquivo utilizado no serviço implementado no CBPF é:

-------- CORTE AQUI ----------
CLEAR = NO **<-- Não apaga a tela**
HANGUP=YES **<-- desliga o modem a cada inicio de secao**
INIT="" ATZ\r OK **<-- espera por nada, envia um RESET e espera por OK.**
WAITFOR=RING **<-- espera o telefone tocar**
CONNECT="" ATA\r CONNECT \s\A **<-- espera por nada, atende, espera por CONNECT e filtra a velocidade.**
-------- CORTE AQUI ----------

4.3.2) Configuração do modem

Como dito anteriormente nossa equipe minimizou bastante o processo de configuração do modem. Essencialmente forçamos a utlização da configuração de fábrica do USRobotics para controle de fluxo por Hardware e ajustamos apenas um grupo de características que nos convinha. Segue-se o procedimento de configuração para um getty configurado de acordo com o descrito na seção 3.1.

É preciso iniciar um emulador de terminal para enviarmos os comandos ao modem. Em praticamente todas as distribuições do Linux é provido um muito similar ao TELIX do MSDOS, o minicom.

Nota: Para maiores informações dos comandos do modem consulte o manual do mesmo.

POCEDIMENTO:

1 - desligue o modem e coloque as microchaves nas seguintes posições:

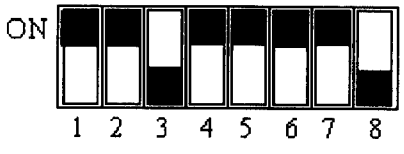


Figure 7 - Microchaves de um MODEM externo

2 - ligue o modem.

3 - Como "root" inicie o minicom;

Se o mesmo já estiver configurado para utilizar o modem, pule para o item 5.

4 - Configure a porta utilizada pelo minicom, para isso tecle <CTRL><A> seguido de <O>, selecione então a opção "Serial port setup" do menu, especifique então a porta (device) correspondente ao modem teclando <A>, especifique então a velocidade do modem teclando <E>, saia de todos os menus teclando <ESC> quantas vezes forem necessárias;

5 - Digite <ENTER> <A> <T> <&> <F> <1> <ENTER>. Isso seleciona o modem para a configuração predefinida de fábrica utilizando controle de fluxo por hardware. O modem então deve responder com um OK, do contrário repita o item 3, se na segunda tentativa o modem ainda não responder OK, o item 2 não foi seguido corretamente ou o modem está com problemas.

6 - Digite <A> <T> <&> <A> <0> <ENTER>. Isso especifica que o modem retornará somente "CONNECT XXXXXX" ao invés de "CONNECT XXXXXX/..." quando completar uma conexão. O modem deve responder com um OK.

8 - Digite <A> <T> <E> <0> <ENTER>. O modem deve responder com um OK. Este comando inibe o "eco" do que lhe e enviado, evitando que o getty se confunda.

9 - Com calma digite <A> <T> <&> <W> <0> <ENTER>. Com o "eco" inibido, você não deve ser capaz de ler o que digita. Este comando grava o estado do modem no Template 0 da NVRAM.

10 - Com calma digite <A> <T> <&> <W> <1> <ENTER>. Este comando grava o estado no Template 1 da NVRAM.

11 - Saia do minicom teclando <CTRL><A> <X>.

Sempre que errar, não será possivel corrigir o erro com <BACKSPACE>, reinicie o procedimento a partir do item 5.

4.4) O PPP Daemon (pppd)

O ppp daemon é um processo que, uma vez ativado, controla uma conexão previamente estabelecida, na forma de terminal (tty device), adotando sobre a mesma a disciplina de linha do protocolo PPP. Uma conexão PPP implica na ativação desse daemon em ambas as máquinas que estão se conectando[8].

O funcionamento efetivo desse daemon varia de sistema para sistema. No caso do Linux, o suporte ao protocolo PPP (isto é, o encapsulamento propriamente dito) na

[8] No caso de uma conexão UNIX <-> UNIX, o PPP Daemon precisa estar ativo em ambas as máquinas. No caso de uma conexão UNIX <-> OUTRO cabe ao outro sistema providenciar um processo de comportamento equivalente ao mencionado daemon. Normalmente, em sistemas não-UNIX, esses processos são programas que têm como finalidade não só "emular" o protocolo PPP como também de estabelecer a ligação telefônica, como é o caso do Trumpet, um cliente PPP para Windows 3.1 sobre o MSDOS.

verdade se encontra no kernel[9] do próprio sistema, assim o PPP daemon é responsável pelo estabelecimento e corte da conexão, i.e., no instante em que é ativado, negocia os parametros do "link" com a máquina remota e, em caso de sucesso, informa ao kernel que o mesmo deve adotar sobre a linha a disciplina do protocolo. Em outros sistemas, o pppd é ainda responsável pelo encapsulamento propriamente dito dos datagramas.

Em todos os casos, o ppp daemon é ainda responsável pela inclusão e exclusão de novas entradas na tabela de roteamento do sistema e dos detalhes intrínsecos associados a esse procedimento como será descrito ainda nesse documento (PROXYARP).

Essencialmente, dada a necessidade da manipulação da tabela rotas e outros parâmetros de configuração dinâmica do sistema, o daemon deve ser um processo pertencente ao "root" ou super-usuario do sistema. Porém, como o propósito mais comum da utilização desse protocolo é permitir "links" de rede por via telefônica e, não obstante, para usuários comuns, estes devem dispor de um meio de ativá-lo sem no entanto possuir os "privilégios" de super-usuario, assim, o pppd é um processo do tipo "setuid"[10].

Serão agora brevemente descritos nessa seção, com maior ou menor ênfase os argumentos de linha de comando do PPP Daemon, pppd (esta descrição se refere a versão corrente no momento da edição desta nota técnica: 2.2.0f). Note que os argumentos de linha podem residir todos ou em parte em arquivos de configuração, de forma a estabelecer as opções default. São esses arquivos: /etc/ppp/options e /etc/ppp/options.ttyname. É possível portanto definir opções default distintas para tty-devices diferentes.

FORMA GERAL:

pppd [tty name] [speed] [options] [options2]

<tty name> - Refere-se ao device de caractere sobre o qual o daemon deve adotar a disciplina de linha ppp, o prefixo "/dev/" é automaticamente acrescentado se necessário o for. Se este argumento for omitido, o daemon atuará sobre o seu tty de controle, i.e, sobre o qual foi chamado (motivo, inclusive, pelo qual deve-se evitar a chamada a "pppd" sem este argumento manualmente a partir de um shell) e fará uma chamada de sistema a função fork() se transladando para execução em "background".

---

[9] **Kernel** é o núcleo do sistema operacional, pode-se entender como sendo o próprio sistema.

10 Um processo **"setuid/setgid"** é um processo capaz de alterar sua propriedade durante sua execução, podendo, portanto, passar a pertencer a outro usuário e/ou grupo "on the fly". Para tanto, é necessário que o arquivo executável pppd (normalmente encontrado no diretorio /sbin ou /usr/sbin) possua o atributo "s". Embora não seja o objetivo desse documento se aprofundar nesta área, é importante esclarecer que o atributo "s" não realiza nenhuma "mágica", o mesmo simplesmente permite que o processo requisite do kernel uma mudança de propriedade, cabe ao processo realizar tal requisição.

<speed> - Define a velocidade com que o daemon se comunicará com o tty-device, note que tal definição nada tem a ver com nenhum procedimento de configuração de um suposto modem, o daemon pode nem mesmo estar "falando" com um modem. Se plausível, a configuração do modem deve ser feita na respectiva configuração do processo "getty" associado à linha. Note porém que se o daemon realmente estiver falando com um modem (/dev/cua?? ou /dev/ttyS??) a UART do mesmo detecta a velocidade e ele (se permitido) realizará um FALL-BACK para esta velocidade se a conexão hover sido estabelecida a uma velocidade superior. Em sistemas como 4.4BSD e NetBSD qualquer velocidade pode ser indicada, em outros sistemas como SunOS apenas um grupo de velocidades é valido. Valores comumente utilizados sao: 9600, 19200, 38400...

<options> - Existe um grande número de opções que podem ser passadas ao pppd, este documento não se compromete a descrevê-las todas ou com a mesma ênfase, mesmo porque algumas delas podem deixar de existir em novas versões do daemon. Especialmente as opções de autenticação não serão mencionadas, se o leitor desejar maiores informações sobre autenticação deve pesquisar os documentos públicos de descrição do protocolo (internet RFC1332). Seguem abaixo as descrições de um subconjunto de opções do pppd:

asyncmap <map> - Define quais caracteres não devem ser diretamente transmitidos através do tty-device, ao invés devem ser convertidos em uma seqüência de código (escape sequence) de dois caracteres em uma ponta para serem reconvertidos a seu valor original na outra. Tal procedimento se deve exclusivamente ao fato de que dependendo do meio ou da forma como se estabelece o "link" alguns caractéres podem ser considerados inválidos. Dois casos clássicos em que isto ocorre são: quando o controle de fluxo do modem utilizado na ligação é feito por software, tal método inclui a transmissão de dois caracteres de controle que evidentemente serão confundidos se enviados com outra intenção; e quando a via de ligação inclue um telnet (isto ocorre quando a máquina que possui o modem não é a real servidora de PPP, "passando" a ligação adiante através de um telnet), assim um caractere "^]" será erroneamente interpretado "no meio do caminho" como um corte da ligação. Em ambos os casos esses caractéres são "proibidos" e devem ser referenciados indiretamente. O uso de asyncmap certamente reduz a velocidade da transmissão quando esses caracteres estão presentes em algum pacote. <map> é um número de 32bits em que cada bit é associado a um caractere "proibido", assim por meio deste argumento é possivel evitar a transmissão de qualquer dos caractéres de controle da tabela ASCII (0-32). É recomendado, com o fim de facilitar o entendimento dessa máscara, se é que é possivel fazê-lo, representá-la em hexadecimal acrescentado o prefixo "0x" à mesma. A máscara utilizada para o controle de fluxo por software é então: 0x000A0000 e a do telnet 0x08000000, a combinação de ambas: 0x080A0000.

connect <p> - Executa o comando de shell <p> antes de iniciar a conexão ppp propriamente dita, esta opção é muito utilizada com o intuito de automatizar o processo de conexão na máquina cliente, normalmente <p> consiste em um shell script que utiliza o programa "chat" (que faz parte da distribuição do ppp daemon) para estabelecer a ligação por modem (incluindo discagem e procedimentos iniciais como automaticamente enviar login e password, etc..).

crtscts - Indica que o pppd deve usar controle de fluxo por Hardware sobre a linha serial durante a conexão, esta opção é comumente utilizada se o meio fisico da transmissão é um modem.

defaultroute - Adiciona a máquina remota à tabela de roteamento do sistema como rota principal de saída (gateway). Esta opção é comumente utilizada do lado do cliente de uma ligação ppp por linha telefônica dado que este geralmente não está realmente conectado a uma rede. Esta nova entrada na tabela de roteamento é automaticamente removida quando a conexão PPP é encerrada.

disconnect <p> - equivalente a opção connect, exceto que <p> é executado tão logo a ligação é encerrada.

escape xx,yy,... - esta opção é semelhante a opção asyncmap, nela também são especificados caractéres que devem ser indiretamente transmitidos, porém aqui como uma lista de numeros hexadecimais individuais cada qual representando um caratere. Diferentemente da opção asyncmap, caractéres cujo código ASCII é maior do que 31 podem ser incluidos na lista.

file <f> - Inclui o conteúdo do arquivo <f> como parte da lista de opções aqui descritas.

lock - Indica ao pppd que este deve criar um "lock-file" no estilo UUCP sobre a linha serial utilizada garantindo somente para si a permissão de utilizá-la. Esta opção é comumente utilizada em links permanentes, deve ser utlizada com MUITA cautela, quando não evitada, em links temporários, pois se a trava não for removida após o uso da linha, o servidor pode tornar o serviço não disponível até que isto seja detectado e resolvido pelo administrador.

mru - (Maximum Receive Unit) Especifica o tamanho máximo permitido para os pacotes de rede que são RECEBIDOS, como explicitado na descrição do protocolo desta nota.

mtu - (Maximum Transfer Unit) Especifica o tamanho máximo permitido para os pacotes de rede que são ENVIADOS.` Após a fase de negociação, o MTU de um lado é igual ao MRU do outro e vice-versa, segundo o critério do menor requisitado em ambos os casos.

netmask - Especifica o NETMASK (Mascara de Rede) utilizada para o device de rede associado a esta conexão PPP (ver nota auxiliar no vinal da seção).

passive - Escolhe o procedimento passivo explicitado na descrição do protocolo PPP/LCP (Link Control Protocol). Se esta opção é utilizada, o daemon tenta estabelecer a conexão, como normalmente o faria, mas no caso de não obter sucesso, aguarda silenciosamente até que um pacote LCP seja recebido da estação remota.

silent - Equivalente à opção passive.

Nota: se ambas as estações utilizarem esta opção, fica reduzida a chance de sucesso na conexão dado que cada lado transmitirá somente uma tentativa de estabelecer contato. A recomendação é que somente a estação "servidora" o utilize.

-ac - Não aceita negociar com a máquina remota com relação a compressão do campo de controle e de endereços (address/control compression).

-all - Não aceita nenhum tipo de negociação com a máquina remota para LCP ou IPCP, i.e., impõe os seus valores.

-am - Não aceita negociação do asyncmap.

-as <n> - o mesmo que asyncmap <n>.

bsdcomp nr.nt - Requer da máquina remota que a mesma envie pacotes utilizando o esquema de compressão BSD. com tamanho máximo de código de "nr" bits e aceita enviar para a mesma pacotes comprimidos com o tamanho máximo de código de "nt" bits. nr e nt são valores compreendidos entre 9 e 15, o valor 0 indica não compressão.

-bsdcomp - Não aceita negociar compressão de pacotes, não requerindo e nem permitindo que pacotes comprimidos sejam enviados ou recebidos.

-crtscts - Desabilita o controle de fluxo por Hardware, se nem crtscts nem -crtscts forem indicados, o esquema de controle de fluxo da linha serial e mantido intacto.

-d - Aumenta o nível de depuração ou "logging" via syslog (consultar a página de manual on-line "man" para configuração do processo syslog).

debug - O mesmo que -d.

-defaultroute - desabilita a opção defaultroute. Esta opção é utilizada para evitar que usuários tentem utilizar o pppd em suas contas (segundo a abordagem de SCRIPT DE ATIVAÇÃO da seção anterior deste documente). Para tal o administrador deve adicionar esta opção no arquivo /etc/ppp/options, que vem a ser o arquivo das opções default do pppd.

-detach - Faz com que o pppd não se translade para background utilizando fork(), pois se assim o fizer em um script segundo as bordagens CONTA GENÉRICA e CONTA INDIVIDUAL o shell script terminará e como o mesmo é na verdade o shell da conta, o "usuário" será "deslogado", encerrando a conexão.

dns-addr <n> - Esta opção existe para atender aos clientes PPP do Windows-NT, pode ser utilizada mais de uma vez, e a ordem estabelece qual a hierarquia de cada DNS.

-ip - Desabilita a negociação de endereços. Se esta opção for utilizada o endereço remoto precisa ser especificado na linha de comando ou no arquivo de configuração. Tal argumento utilizando na estação servidora impede que uma máquina cliente tente se auto-designar um endereço.

[+/-]ip-protocol - Habilita/desabilita o uso dos protocolos IPCP e IP. A condição default é habilitado. Esta opção deve somente ser utilizada se a estação cliente for somente capaz de entender o protocolo IPX.

[+/-]ipx-protocol - Equivalente a opção descrita acima, porém refere-se ao protocolo IPX.

ipcp-accept-local - Esta opção faz com que a máquina local aceite o endereço "sugerido" pela estação remota como o seu próprio endereço para o link mesmo que em discrepância com o endereço especificado na linha de comando.

ipcp-accept-remote - Equivalente a opção descrita acima, porém refere-se ao endereço da estação remota.

Nota: É recomendavel que a estação servidora tenha como um dos seus parametros "-ip" e as clientes incluam em sua lista de parametros "ipcp-accept-local" e "ipcp-accept-remote".

local - Esta opção indica ao pppd que o mesmo deve tratar o device como não sendo um modem, implicando que o mesmo não levará em consideração o sinal de CARRIER-DETECT e não mudará o estado do sinal DTR (Data Terminal Ready).

modem - O oposto da opção local. O uso desta opção (default) indica ao pppd que o mesmo deve aguardar o sinal de CARRIER DETECT como forma de detecção de que a ligação telefônica foi completada. Ainda o pppd momentaneamente desligará brevemente o sinal DTR imediatamente antes de executar o script de conexão e após o fim da mesma como forma de forcar o modem a "fechar" (HANGUP) qualquer ligação corrente nesses instantes.

Nota: Esta opção é recomendada em ambas as estações, servidora e cliente, como forma de evitar que a linha fique "trancada" sem uso.

-mn - Desabilita a identificação do tipo de device por meio de números mágicos (uma especie de número de identidade que classifica o tipo de periférico de um tty-device). Esta opção evita que o pppd detecte uma linha em "loop" e pode ser útil durante a fase de implementação.

-[mru/mtu] - Desabilita a negociação do MRU ou do MTU.

-p - O mesmo que a opção passive.

-pc - Desabilita a negociação da compressão do campo de descrição do protocolo encapsulado.

persist - Faz com que ao fim de uma conexão o pppd tente reestabelecê-la ao invés de terminar sua execução.

proxyarp - Adiciona uma entrada na tabela de resolução de endereços (ARP) associando o endereço IP da máquina remota com o endereço ethernet da máquina local. Esta opção será detalhadamente discutida em uma seção posterior deste documento.

-proxyarp - Desabilita e impede que seja alterada a tabela ARP do sistema. Como um usuário qualquer (No caso do modelo de Script de Ativação) pode executar o pppd com seus próprios argumentos, e o uso de proxyarp pode ser indesejado, o administrador do sistema pode usar esta opção no(s) arquivo(s) de configuração impedindo o seu uso.

-vj - Desabilita a negociação da compressão dos cabeçalhos do protocolo TCP-IP.

xonxoff - O oposto de crtscts. Estabelece o controle de fluxo por software.

NOTAS AUXILIARES DA SEÇÃO 4:

Em alguns sistemas UNIX, os devices de rede residem sob o diretóro /dev como os demais, normalmente como os nomes eth0, en0, ppp0, slip0, eth1... Particularmente no Linux, estes devices não estão presentes em nenhum filesystem o real, e sim no pseudo-filesystem "proc" e são criados e destruídos dinamicamente. A boa norma de programação UNIX estabelece que todos os processos devem se comunicar com estes devices através de "sockets" (sob os "olhos" do kernel) e não diretamente. Os devices de rede ppp atendem aos nomes ppp0, ppp1, ... assim como os ethernet a eth0, eth1, ...

A Tabela de roteamento do sistema mapeia por qual device deve ser enviado cada grupo de endereços, permitindo assim que uma estação funcione como um roteador se adequadamente configurada.

Quando uma conexão PPP e estabelecida, a mesma somente é utilizada se alguma rota assim o indicar. Existe o conceito na citada tabela de rota padrão (default), i.e., sempre que um pacote tem um destino não muito claramente evidente na tabela de roteamento, ele e enviado para esta rota.

Para maiores informações sobre Administração de Rede e compreensão dos conceitos de Netmask, Broadcast address, Network address, etc, o leitor deve consultar o manual "Linux Network Administration Guide" do projeto Linux Documentation Project (LDP), consultar bibliografia para obtenção do mesmo.

4.5) Criando uma conta PPP

A criação de uma conta PPP se assemelha a de uma conta normal, porém deve-se evitar a utilização de scripts de criação como o "addusr" ou "adduser", pois estes automaticamente constroem o diretório do usuário e tomam uma série de providências que quando não desnecessárias são indesejáveis para este proposito. A princípio o ideal seria alterar o "adduser" criando um outro script, como por exemplo "addppp" ou "addpuser". Neste documento descreveremos o procedimento manual para criação de contas.

4.5.1) Criando uma conta com IP estático

O primeiro passo é adicionar uma entrada no arquivo de senhas "/etc/passwd" seguindo este padrão (recomenda-se o uso do comando "vipw" para a edição deste arquivo):

---------- CORTE AQUI ----------

p*usuário*::UID:GID:PPPstatic user *usuário*:/tmp:/sbin/pppstatic*usuário*

---------- CORTE AQUI ----------

Substituindo-se toda ocorência da palavra usuário pelo nome do mesmo, UID pelo primeiro UserID livre e GID pelo grupo (É expressamente recomendada a criação de um grupo exclusivo para as contas PPP de usuários).

Essencialmente isto define um usuário cujo diretório "home" é o diretório temporário do sistema e cujo shell é o script /sbin/pppstatic*usuário*.

O segundo passo é a criação do script propriamente dito que deve ter a forma:

---------- CORTE AQUI ----------

```
#!/bin/sh
mesg n
stty -echo
exec /usr/sbin/pppd 152.84.253.32:152.84.253.249 -detach silent modem crtscts
```

---------- CORTE AQUI ----------

Não se esqueça de mudar as propriedades do arquivo para este usuário/grupo e seus atributos para "-r-x------".

Assim toda vez que este usuário se logar, o sistema automaticamente executa esse script, que por sua vez inicia o pppdaemon para esta linha com o IP apropriado.

4.5.2) Criando uma conta com IP dinâmico

O primeiro passo é semelhante ao do IP estático, porém todas as entradas do "/etc/passwd" referentes as contas dinâmicas utilizam o mesmo script como shell:

---------- CORTE AQUI ----------

p*usuário*::UID:GID:PPPdynamic user *usuário*:/tmp:/sbin/pppdynamic
---------- CORTE AQUI ----------

A criação do script dá-se uma só vez para todos os usuários dinâmicos, o script sugerido é:
---------- CORTE AQUI ----------
```
#!/bin/sh
mesg n
stty -echo
exec /usr/sbin/pppd 152.84.253.32: -detach silent modem crtscts proxyarp
```
---------- CORTE AQUI ----------

É necessário ainda criar uma única vez para cada linha ligada ao sistema um arquivo de configuração do PPP Daemon que as associa aos seus respectivos IPs. O formato do arquivo é simplesmente:

ENDEREÇO_DO_SERVIDOR:ENDEREÇO_DO_CLIENTE

Em todos os arquivos (/etc/ppp/options.<ttydevice>) o ENDEREÇO_DO_SERVIDOR é sempre o mesmo e corresponde ao endereço da estação servidora, como o nome sugere. Já o ENDEREÇO_DO_CLIENTE é único para cada tty e representa o IP associado a cada linha.

Exemplo: /etc/ppp/options.ttyS0

---------- CORTE AQUI ----------
152.84.253.32:152.84.253.201
---------- CORTE AQUI ----------

4.6) Problemas de roteamento e soluções (PROXYARP).

O Último problema, com o qual nos deparamos, e o filosoficamente mais complicado é o roteamento.

Quando uma conexão PPP é estabelecida com sucesso, ambas as máquinas ligadas entre si adicionam em suas tabelas de roteamento a outra. O problema é que se a estação servidora não for também o roteador da rede local, nenhuma outra máquina além dela saberá da existência, ou do caminho para se chegar a estação cliente. Existem 4 soluções para este problema.

1 - Tornar a estação servidora o roteador. Esta solução é consideravelmente ruim, além de envolver uma mudança global na configuração da rede local, uma estação raramente é mais eficiente do que um roteador verdadeiro;

2 - Adicionar manualmente à tabela do roteador a estação servidora como caminho para todos os IPs das estações cliente. Esta solução também é trabalhosa e de difícil manutenção (principalmente quando usados IPs estáticos).

3 - Utilizar um daemon de "**broadcasting**" de rotas como **gated** ou **routed**. Esses daemons são capazes de informar o roteador do surgimento e desaparecimento de novas rotas dinamicamente. Esta opção é uma automatização da opção 2. Porém esses daemons são difíceis de se configurar, são incompatíveis com alguns roteadores (que não seguem a risca o protocolo ARP (address Resolution Protocol), geram tráfego desnecessário de rede e uma série de outras implicações que aumentam a complexidade do problema.

4 - Utilizar PROXYARP (também chamado de ARP elegante) que é uma solução simples e eficiente.

PROXYARP

Toda interface ETHERNET possui um endereço ETH único. O papel do roteador é criar uma associação entre endereços IP e endereços ETH. Dentro de uma rede local os pacotes TCP/IP são envelopados e etiquetados com os endereços ETH, quando o roteador recebe um pacote (que chegou a ele de acordo com a etiqueta ETH) ele o abre, consulta a etiqueta IP, consulta a tabela de associação IP <-> ETH, envelopa novamente o pacote com o endereço ETH da máquina destino e o reenvia à interface correta. Quando uma estação utiliza PROXYARP, efetivamente, ela passa a possuir mais de um endereço IP, porém todos associados ao mesmo endereço ETH.

Em poucas palavras: quando a estação servidora se utiliza de proxyarp ela passa a entender que pacotes enviados para o IP do cliente devem ser tratados por ela, sem PROXYARP esses pacotes sao ingnorados e perdidos.

A atitude prática a ser tomada é: se a estação servidora é o roteador, não se deve utilizar PROXYARP, do contrário PROXYARP é a solução mais prática e elegante, basta adicionar o argumento "proxyarp" a linha de comando do pppd nos scripts da estação servidora.

7 - BIBLIOGRAFIA

- Linux Network Administration Guide
LDP - Linux documentation Project
http://www.linux.org

- Comunicação de Dados e Sistemas de Teleprocessamento
Silveira, Jorge Luis da
Makron - Mcgraw-Hill, 1991

- pppd(8) getty(1), gettydefs(5)
Online manual pages.

- RFC1144
Jacobson, V. Compressing TCP/IP headers for low-speed serial links. 1990 February.

- RFC1321
Rivest, R. The MD5 Message-Digest Algorithm. 1992 April.

- RFC1332
McGregor, G. PPP Internet Protocol Control Protocol (IPCP). 1992 May.

- RFC1334
Lloyd, B.; Simpson, W.A. PPP authentication protocols. 1992 October.

- RFC1548
Simpson, W.A. The Point-to-Point Protocol (PPP).

- RFC1549
Simpson, W.A. PPP in HDLC Framing. 1993 December

NOTAS TÉCNICAS é uma pré-publicação de trabalhos técnicos-científicos do CBPF.
Pedidos de cópias desta publicação devem ser enviados aos autores ou ao:

Centro Brasileiro de Pesquisas Físicas
Área de Publicações
Rua Dr. Xavier Sigaud, 150 – 4º andar
22290-180 – Rio de Janeiro, RJ
Brasil

NOTAS TÉCNICAS is a preprint of technical reports from CBPF.
Requests for copies of these reports should be addressed to:

Centro Brasileiro de Pesquisas Físicas
Área de Publicações
Rua Dr. Xavier Sigaud, 150 – 4º andar
22290-180 – Rio de Janeiro, RJ
Brazil