UFSC – Universidade Federal de Santa Catarina
SETREM – Sociedade Educacional de Três de Maio
CPGCC – Curso de Pós-Graduação em Ciências da Computação

Avaliação de interfaces de comunicação USB em CLUSTER de computadores.

Dissertação submetida ao Programa de Pós-Graduação em Ciências da Computação
para obtenção do grau de Mestre em Ciências da Computação

Fauzi de Moraes Shubeita

Orientador: Prof. Luiz Fernando Friedrich Dr.

Florianópolis, Abril de 2003.

AGRADECIMENTOS

Apesar do meu pai e da minha mãe não saberem muito bem o que é um Mestrado, inicio a dedicatória a eles. Valeu !!! Ao pessoal da SETREM que me deu aquela força, aos amigos e colegas professores além é claro, dos alunos.

A Ana Paula, minha esposa e a minha filha Agatha e ao pessoal de apoio, Léa (sogrinha), Léo, Bisa Altiva, Fernanda, Luísa e Guilherme. Ao meu sogro "Chico" do Canto, que nessas horas deve de estar no seu rancho ao pé da serra, lá na Boca do Monte, no município de Santa Maria.

Também ao meu Irmão Irsan, ao qual manteve o fornecimento de tinta e impressoras que tornaram possível confeccionar essas páginas (valeu mano), a cunhada Bethânia e aos "anjinhos" Ihana e Yasser. Ao meu irmão Samir pelas longas conversas sobre as "coisas do mundo" e as irmãs Sissa e Zakie.

Agradecimentos aos meus colegas de mestrado, em especial a Andréa Bordin pela "mão" dada no início e final do curso e a todos que até agora estão esperando o "churasco".

Enfim, a todas as pessoas que mesmo com os nomes não relacionados nessa página fizeram parte dessa etapa tão importante na minha formação profissional e a DEUS, que sempre esteve presente. Obrigado a todos.

ABSTRACT

To computacional of loud performance always fascinated the tester of hardware. To find incessant for performance drove several projects of computacional cooperative, in the scenery the CLUSTER gathered place of prominence.

That's work will be approached him subject matter CLUSTER its point more weak: the interfaces of connection. In the chapter 2 is presented the models conceptual of hardware in systems distributed and the respective models set na Taxonomia of Flynn.

Him surrender 3 explores the leading technology in protocols of communication internet and the of drop latency, as well as him hardware necessary for to assembly and shape of cenaries of multicomputers. Him surrender 4 mentions the nets of interconexion and your types.

Him surrender 5 is presented him Multicomputer CRUX, to its architecture of operation and of hardware. The surrender 6 is presented him bar USB and all the your characteristics techniques, to form as to communication serial works in USB and as him Linux undertakes the packs USB-Ethernet.

The Chapter 7 presents the offsprings of performance after the tests with the periferal USB and the sceneries of implementacion in the Multicomputer CRUX conecting for USB.

Through of analysis and tests of performance, can argue that the connections in net to make user of connections PCI in machines heterogeneous can be one solution of low cost.

Him standard USB, with its flexibility and conection, will can substitute the current plates of nets Ethernet and Fast-Ethernet set in the PCI bar, in CLUSTER machines.

RESUMO

A computação de alto desempenho sempre fascinou os experimentadores de hardware. A busca incessante por performance conduziu diversos projetos de computação cooperativa, e nesse cenário os CLUSTER obtiveram lugar de destaque.

Nesse trabalho será abordado o assunto CLUSTER no seu ponto mais fraco: as interfaces de conexão. No capítulo 2 é apresentado os modelos conceituais de hardware em sistemas distribuídos e os respectivos modelos baseados na Taxonomia de Flynn.

O capitulo 3 explora as principais tecnologia em protocolos de comunicação internet e os de baixa latência, bem como o hardware necessário para a montagem e configuração de ambientes de multicomputadores. O capitulo 4 cita as redes de interconexão e seus tipos.

No capitulo 5 é apresentado o Multicomputador CRUX, a sua arquitetura de funcionamento e de hardware. No capitulo 6 é apresentado o barramento USB e todas as suas características técnicas, a forma como a comunicação serial funciona no USB e como o Linux trata os pacotes USB-Ethernet.

O capítulo 7 apresenta os resultados de desempenho após os testes com os periféricos USB e os cenários de implementação do Multicomputador CRUX ligado por USB.

Através da análise de testes de desempenho, podemos argumentar que as conexões em rede utilizando-se de ligações PCI em máquinas heterogêneas pode ser uma solução viável e de baixo custo. Porém o padrão USB, com sua flexibilidade e conectividade, poderá substituir as atuais placas de redes Ethernet e Fast-Ethernet baseadas no barramento PCI, em vários outros ambientes CLUSTER.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

Nos sistemas computacionais atuais, os fatores mais relevantes são o desempenho do hardware e do software. Melhor projeto dos sistemas, uso dos benefícios da multitarefa e otimização de código são expressões que fazem parte do dia-a-dia dos desenvolvedores, mas ainda assim, nem sempre são o suficiente para garantir o desempenho necessário.

Para a grande maioria desses casos existem os supercomputadores. Porém, em função do seu alto custo de manutenção e implementação, além de outros custos agregados, eles estão muito além do poder aquisitivo de muitas instituições.

A utilização de Clusters são uma alternativa para se conseguir alto poder de processamento com baixo custo, ou seja, o uso de vários computadores trabalhando em conjunto para a resolução de problemas.

Podemos implementar Clusters usando as plataformas PC atuais que proporcionam alto poder de processamento aliado ao baixo custo financeiro e operacional, porém não é nenhuma novidade que o ambiente de rede utilizado é um obstáculo nos ganhos de performance.

Os projetos atuais de sistemas computacionais paralelos são essenciais em ambientes que utilizam alto poder de processamento de dados. Porém, esses sistemas sofrem com a falta de desempenho nos protocolos de comunicação entre as máquinas interligadas, prejudicando bastante todos os processos de envio e recebimento de dados.

Em arquiteturas do tipo CLUSTER, a comunicação entre maquinas é normalmente suportada por protocolo TCP/IP e adaptadoras de rede Ethernet ou Fast-Ethernet.

Esse trabalho tem como objetivo explorar a performance do barramento USB 1.1, propondo-o como meio físico de comunicação em ambientes computacionais do tipo CLUSTER em substituição das adaptadoras do padrão Ethernet.

Esta organizado de uma forma que contemple os aspectos de software e hardware envolvidos na construção e funcionamento do mesmo em ambiente operacional Linux.

O modelo seguindo foi o CRUX [COR1996], atualmente implementado junto aos Laboratório de Computação Paralela e Distribuída da UFSC.

A interface USB em substituição as adaptadoras padrão Ethernet e/ou Fast-Ethernet baseados no barramento PCI, pode ser uma alternativa para a comunicação entre os elementos do CLUSTER, propondo possíveis cenários de implementação.

Os cenários apresentados procuram explorar os aspectos de desempenho aliado a baixo custo como justificativa viável para a utilização do barramento USB. A evolução do barramento USB e o interesse da indústria em desenvolver novos dispositivos baseados em USB justificam a viabilidade do projeto realizado.

2 MODELOS CONCEITUAIS DE HARDWARE EM SISTEMAS DISTRIBUÍDOS

A busca por incremento de performance nos computadores é um desafio para desenvolvedores de softwares e construtores de hardware. Porém, nunca sabemos com certeza qual a influência real que um ou outro exerce sobre um sistema computacional até fazermos uma análise detalhada de ambos.

Podemos também fazer uma divisão dos diversos modelos de computação paralela contemplando os seguintes pontos:

Modelos para aplicação: Servem para representar o paralelismo potencial de um algoritmo;

Modelo de Máquina: Descrevem as principais características da máquina paralela, como o fluxo de instruções e a comunicação entre os processadores;

Modelos de execução: Detalham as formas de programação e seus paradigmas.

A seguir estão relacionados as principais arquiteturas de hardware que atualmente fazem parte do cenário computacional dos sistemas distribuídos seguindo a Taxonomia de Flynn [Fly66], que classifica os computadores através do fluxo de instruções (Instruction ou I) e de dados (Data ou D). Esses podem ser únicos (Single ou S) ou múltiplos (Multiple ou M).

2.1 TAXONOMIA DE FLYNN

É muito difícil a tarefa de classificar computadores paralelos. Já foram feitas diversas sugestões e a classificação definitiva ainda esta por vir. Porém, a que trouxe melhores resultados e ainda hoje é usada, é a proposta por Flynn.

Hoje em dia a maioria das máquinas paralelas é MIMD, logo a classificação de Flynn não é mais suficiente. Essa classificação esta baseada em dois conceitos: **fluxo de instruções e fluxo de dados**.

O **fluxo de instruções** esta relacionado com o programa que o processador executa, enquanto que o **fluxo de dados** esta relacionado com os operandos manipulados por essas instruções. O fluxo de instruções e o fluxo de dados são

considerados independentes e por isso existem quatro combinações possíveis, relacionadas abaixo.

		Fluxo de Instruções	
		Serial (SI)	Paralelo (MI)
Fluxo de Dados (D)	Serial ou único (SD)	SISD	MISD
	Paralelo ou múltiplo (MD)	SIMD	MIMD

FIGURA 1- POSSÍVEIS COMBINAÇÕES DE FLUXO DE INSTRUÇÕES E FLUXO DE DADOS

2.1.1 SISD

Computador com um fluxo de instruções e um fluxo de dados (todos os computadores com um único processador) e executam as instruções de forma serial. Podemos ter mainframes com mais de um processador, mas cada um deles executa as instruções de forma independente. Também é a arquitetura usada nos computadores em nossos lares e segue o modelo proposto por Von Neumann.

2.1.2 SIMD

Computador com um fluxo de instruções e múltiplos fluxos de dados. Maquinas que possuem 8, 16, 32, 1024 ou até mais unidades de processamento que executam as instruções com diferentes dados em Lock-step. Isso significa que todos os processadores trabalham sincronizados e executam todos a mesma instrução. Vem daí a definição de computador vetorial.

2.1.3 MISD

Computadores com múltiplos fluxos de instruções e um fluxo de dados. É um conceito muito raro de se aplicar e implementar, tanto que é um assunto bastante divergente entre

os pesquisadores. Porém, nas leituras especializadas utiliza-se esse conceito para definir o Pipeline (linha de produção) nos computadores.

2.1.4 MIMD

Múltiplos fluxos de instruções e múltiplos fluxos de dados. Sistemas distribuídos são considerados MIMD Essas máquinas podem ser divididas em dois grupos: aquelas com memória compartilhada, chamadas multiprocessadores e aquelas com memória privativa chamada de multicomputadores.

2.2 ARQUITETURAS MIMD

Normalmente dividimos os modelos MIMD em Multi-computadores e Multi-processadores [TAN1997], sendo que as latências para a troca de mensagens é maior nos Multicomputadores. Também podemos destacar a arquitetura de memória, que pode ser dividida da seguinte forma:

UMA: (*Uniform Memory Access*) Existe um espaço de endereçamento comum acessível a todos os processadores. O tempo de acesso a esta memória é aproximadamente o mesmo para todos os processadores.

NUMA (*Non-Uniform Memory Access*) Cada processador possui a sua memória local. Nesse caso, a máquina e seu sistema operacional fornecem primitivas de comunicação entre os processadores. Existem diversas formas de acesso a memória de outros processadores, que pode ser um espaço de endereçamento único ou então troca de mensagens. Em máquinas com arquitetura NUMA, o tempo de acesso a uma memória não local é muito maior que do acesso local. A noção de localidade é muito importante.

Além delas podemos destacar:
NORMA - Sem acesso à memória remota
NCC-NUMA - Sem acesso uniforme à memória e sem coerência de cache.
CC-NUMA - Sem acesso uniforme à memória e com coerência de cache.
COMA - Arquitetura de memória somente de cache.
SC-NUMA - Sem acesso uniforme à memória, com coerência feita por software.
Essas definições não foram exploradas a fundo nesse projeto, sendo apenas citadas.

2.2.1 MULTICOMPUTADOR

O modelo de máquina paralela, chamado multicomputador é composto de vários computadores de *von Neumann*[2], que são os nós, ligados por uma rede de interconexão. Cada computador executa seu próprio programa. Este programa pode acessar uma memória local e pode enviar e receber mensagens pela rede. Mensagens são utilizadas para comunicação com outros computadores ou, equivalentemente, a ler e escrever em memórias remotas. Na rede ideal, o custo de enviar uma mensagem entre dois nós é independente tanto da localização do nó quanto de outros tráfegos na rede, porém depende do tamanho da mensagem.

Um atributo que define o modelo de multicomputador é que acessos a **memória local** (mesmo nó) são mais baratos que **acessos remotos a memória** (nó diferente). Ou seja, ler e escrever são menos custosos que enviar e receber. Portanto, é desejável que acessos a dados locais sejam mais freqüentes que acessos a dados remotos. Esta propriedade chamada **localidade**, é um requisito fundamental em software paralelo, além de concorrência e escalabilidade. A importância da localidade depende da relação entre o **custo de acesso remoto e local**. Esta relação pode variar de 10:1 a 1000:1 ou mais, dependendo da performance relativa do computador local, da rede, e dos mecanismos utilizados para mover os dados de e para a rede.

2.2.2 MULTIPROCESSADOR

Um computador paralelo é um conjunto de processadores capazes de trabalhar cooperativamente para resolver um problema. A utilização de computação de alto desempenho tem sido motivada não só por simulações numéricas de sistemas complexos (tempo, circuitos eletrônicos...), mas atualmente também por aplicações comerciais onde há uma grande quantidade de dados para serem processados de maneiras sofisticadas.

Essas aplicações, segundo Foster [FOS1995], incluem vídeo conferência, diagnósticos médicos auxiliados por computador, bancos de dados paralelos, realidade virtual, computação gráfica, dentre outras. Em um ambiente paralelo para o desenvolvimento de algoritmos há necessidade de *hardware* e *software* adequados para esse tipo de processamento..

2.2.3 CLUSTER DE PC

Apesar de ser um Multicomputador, a adoção de Clusters, coleção de estações de trabalho/PC conectadas por uma rede local, tem virtualmente explodido desde a implementação do primeiro *Beowulf* em 1994. Há uma atração criada no potencialmente baixo custo não só de hardware mas também de software, mas também como o controle que os construtores/usuários tem sobre esses sistemas.

O interesse por Clusters pode ser visto, por exemplo, pela atividade do IEEE Task Force on Cluster Computing (TFCC) que regulamenta as publicações e informa sobre as atualizações da computação em Cluster. A quantidade e diversidade de soluções oferecidas pelas mais diversas empresas vendedoras demonstram uma crescente demanda por esse tipo de solução.

Um *cluster* é um tipo de sistema paralelo ou distribuído que consiste de pequenas máquinas agregadas, utilizadas como um recurso de computação único [PFI998]. Existem algumas vantagens na utilização de CLUSTER, como por exemplo, o desempenho que pode ser obtido, baixo custo em relação às máquinas paralelas dedicadas, facilidade de expansão e *software* sem custos. Arquiteturas baseadas em clusters podem ser homogêneas (todos os nodos que a compõem possuem a mesma arquitetura e sistema operacional) ou heterogêneas (os nodos possuem processadores diferentes e/ou sistemas operacionais diferentes).

2.2.4 TAMANHO DO GRÃO

O custo na comunicação entre os vários NÖS pode ser um gargalo na concepção de algoritmos paralelos. Entende-se como comunicação, desde o simples fato de ter uma única memória que é acessada por vários NÖS (e ai aplicam-se bloqueios) ou o custo do envio de mensagens.

Mesmo em memória compartilhada, pode não compensar fazer a distribuição do código a nível de **instrução**, entregando uma instrução de baixo nível para um NÓ e outra para um segundo NÓ. Como elas poderão disputar a mesma região de memória, pode ser desastroso.

Então, pode ser que um maior desempenho seja alcançado se entregar, por exemplo, uma **função** para um NÓ e outra função para o segundo. Talvez, devido aos altos custos de uma máquina sem memória global, somente a distribuição de **processos** inteiros compense, pois eles não irão comunicar-se tanto[BAR1998].

A isto dá-se o nome de *grão*, ou seja, o quão fino será o processamento. Grãos menores aproveitam melhor o processamento, mas podem ser lentos. Grãos maiores podem resolver o problema do custo de comunicação, mas pode fazer com que um NÓ termine muito antes que os outros, ficando ocioso. Em máquinas com memória compartilhada, procura-se usar uma granulosidade mais fina, hoje com o emprego de threads, enquanto que em máquinas sem esta memória, deve-se usar uma mais grossa, talvez a nível de processos inteiros[BAR1998].

2.3 MODELOS DE EXECUÇÃO

O modelo de execução esta profundamente relacionado com o modelo de máquina, pois eles são a base da programação de uma máquina paralela. Eles fornecem a semântica da execução. Um dos seus principais objetivos é fornecer boas previsões do tempo de execução de um programam paralelo.

MIMD: (Multiple Instrution Multiple Data) Modelo com memória distribuída (entre os nodos) e memória compartilhada entre nodos. Dessa forma, para a programação paralela utiliza-se trocas de mensagens para haver a comunicação entre os nodos e threads para usufruir da capacidade de n processadores que compartilham a memória nos nodos.

SPMD :*(Single Program Multiple Dat*a), ou seja, o mesmo processo é disparado simultaneamente em todos os nodos. Como parâmetros de inicialização, o processo recebe o número do nodo onde foi disparado, o número total de nodos e os endereços IP desses nodos. O número de cada nodo é dado pela ordem em que aparece na lista: para p nodos, a numeração varia de 0 a p-1.

SMP: *(Symmetric MultiProcessing)***,** ou seja, duas ou mais CPU's num único nó compartilhando um mesmo módulo de memória. É um modelo de execução que oferece ganhos de performance quando os algoritmos são implementados e executados seguindo

esse modelo, e do contrário, as vantagens sobre máquinas com um processador não chega a ser significativa, sem contar o seu alto custo [BAR1998].

SMT : *(Simultaneous MultiThreading)***.** São processadores que possuem duas ou mais unidades de execução num mesmo encapsulamento. A grosso modo, seria como ter um SMP num único chip.

É uma nova tecnologia que esta sendo desenvolvida e que permite a CPU executar duas ou mais seções (thread) de um programa, simultaneamente. Para se aproveitar dessa capacidade, as aplicações devem ser especialmente elaboradas.

Uma diferença entre SMT para SMP, é que esse último pode executar várias aplicações simultaneamente com vantagens sobre os SMT ou configurações uniprocessadas. Porém, há uma série de fatores como acesso a memória e divisão de recursos que elevam as diferenças entre SMT e SMP [PC2002].

2.4 REDES DE INTERCONEXÃO

Redes de interconexão são constituídas de entidades de hardware (canais de comunicação) e software (controle de estabelecimento de canais) que são projetadas para facilitar a comunicação entre processos e processadores [MER96].

Para a implantação de da Computação Distribuída é fundamental uma arquitetura de rede robusta e eficiente, que seja confiável para a transmissão das informações e suporte o sistema e as aplicações instaladas. FEN[1981]. Podemos ter quatro decisões de projeto acerca das redes de interconexão:

- **Modo de Operação**: Pode ser síncrona ou assíncrona. Um sistema distribuído deve prover suporte aos dois modos de operação.

- **Estratégia de Controle**: O controle da rede de interconexão pode ser feita através de um único elemento, numa estratégia centralizada, ou através de alguma estratégia distribuída.

- **Metodologia de Chaveamento**: As principais metodologias de chaveamento são por pacote ou por circuito. Em geral o chaveamento de pacotes é mais eficiente para mensagens curtas e o chaveamento por circuito para mensagens muito grandes.

- **Topologia da Rede**: A topologia de rede de interconexão é um fator determinante no projeto de um sistema distribuído. As topologias regulares podem ser estáticas ou dinâmicas.

Existem dois tipos de topologias que podem ser implementadas em redes de computadores: estáticas e dinâmicas.

Estáticas – Os elementos da rede são conectadados por ligações ponto-a-ponto fixas, que não mudam durante a execução.

Dinâmicas – É formado por canais chaveados que são configurados dinamicamente, conforme a demanda do programa em execução.

Além disso, devemos lembrar que o desempenho de uma rede de interconexão depende de 4 fatores: Funcionalidade, Largura de Banda, Complexidade de Hardware e Escalabilidade [BAR1998].

3 MECANISMOS DE COMUNICAÇÃO

Para implementarmos ambientes computacionais distribuídos necessitamos de uma série de elementos de software e hardware. Dentre os elementos de software temos o Sistema Operacional e os Protocolos de comunicação e controle.

Os elementos de hardware envolvem adaptadoras de rede, Switch's, Hub's, Pontes, Roteadores e conectores de diferentes tecnologias. A combinação de bons elementos de Hardware e Software é o que garante a funcionalidade, estabilidade e desempenho de qualquer ambiente computacional paralelo e distribuído.

3.1 PROTOCOLOS DE COMUNICAÇÃO

Os protocolos de comunicação em redes de computadores podem ser dos mais variados, dependendo da sua aplicabilidade, meios de transmissão, hardware conectado, etc... Porém, quando o objetivo é interligar vários PC para que trabalhem juntos, devemos de fazer uma análise detalhada da estrutura desses protocolos e verificar qual o seu impacto no desempenho geral do sistema [TAN1997].

Um protocolo de comunicação define-se por regras e convenções que podem ser utilizadas por dois ou mais computadores em rede para trocar informações. Protocolos podem ser definidos como:

- Orientados a conexão. Oferecer vários níveis de confiança incluindo garantias para ordem de chegada dos pacotes ou sem garantia (com confirmação ou sem confirmação).
- Não-bufferizado (síncrono) ou bufferizado (assíncrono)
- Pelo número de dados intermediários copiados entre buffers, que podem ser zero, um ou mais.

Diversos protocolos são usados em CLUSTER. Protocolos de redes que são desenhados para o uso na Internet e protocolos que são usados na comunicação CLUSTER. Esses, dois tipos protocolos estão sendo designados para o uso na computação CLUSTER, o protocolo Internet e os de baixa latência.

3.1.2 PROTOCOLOS INTERNET

O protocolo Internet (IP) é um padrão para redes Web. O IP oferece um melhor (e duvidoso) serviço de mensagens entre dois computadores que tem um endereço IP. O TCP e o UDP estão na camada de Transporte do IP. O TCP (ou TCP/IP) oferecem realmente um serviço orientado a conexão entre dois Hosts em uma Rede. O UDP é não-confiável fazendo a conexão a nível de Transporte. TCP e UDP, o padrão BSD Sockets API para TCP e UDP foram as primeiras bibliotecas de mensagens usadas na computação Cluster.

Tradicionalmente o TCP e UDP são protocolos tipicamente implementados usando um ou mais Buffers no sistema de memória tendo a ajuda do Sistema Operacional. Para enviar uma mensagem, uma aplicação constrói a mensagem na memória do Nodo usuário, e então faz com que o Sistema Operacional requisite uma cópia da mensagem para o seu Buffer. Uma interrupção do sistema é requisitada para essa intervenção do S.O antes que a mensagem seja enviada para a rede. Quando a mensagem é recebida pelo hardware da rede, ela copia a mensagem para o sistema de memória. Uma interrupção é usada para copiar a mensagens da memória do sistema para a memória do usuário e notifica o recebimento do processo e que a mensagem chegou [TAN1997].

O Overhead, tanto do S.O quanto da cópia da mensagem para o Sistema de memória são porções significativas do tempo total de envio de mensagens. O hardware da rede ficou mais rápido nos anos 90 e o overhead nos protocolos de comunicação tornaram-se significativamente maior do que o hardware atual no tempo de transmissão das mensagens.

3.1.2 ARQUITETURA TCP/Ipv4

TCP/IP é o nome que se dá a toda família de protocolos utilizados pela Internet. Oficialmente essa família de protocolos é chamada Protocolo Internet TCP/IP ou comumente referenciado de TCP/IP [TAN1997].

O diagrama abaixo descreve as camadas do protocolo TCP/IP e seus princiapis protocolos de comunicação. Todos os níveis comunicam-se com a camada imediatamente superior a sua (ver Figura 3).

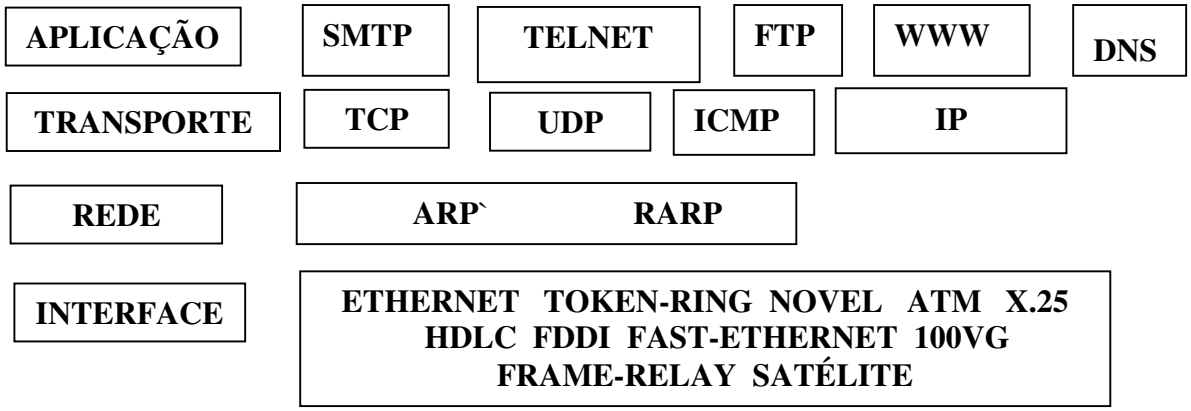

FIGURA 2 –CAMADAS TCP/IPV4.

3.1.3 COMUNICAÇÃO ENTRE CAMADAS – IPv4

Quando fazemos referência a comunicação entre camadas na pilha de protocolos, devemos descrever como ela se processa e o que cada uma das camadas afeta o desempenho extra que é adicionado a execução de qualquer tarefa num ambiente Multicomputador.

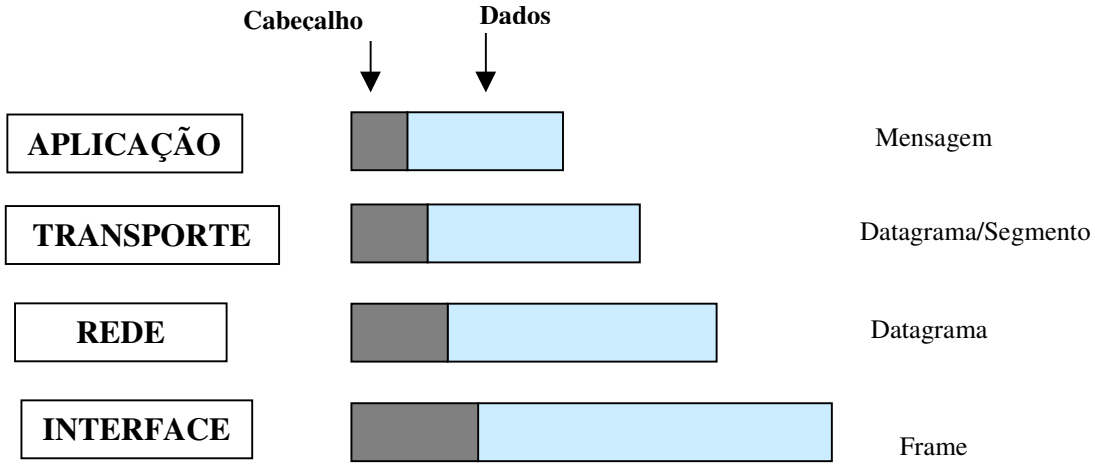

FIGURA 3 – COMUNICAÇÃO ENTRE CAMADAS TCP/IPV4

Camada de Interface: Interface que compatibiliza a tecnologia específica da rede com o protocolo IP, permitindo também que qualquer rede pode ser ligada através de um driver que permita encapsular datagramas IP e envia-los através de uma rede específica, traduzindo os endereços IP lógicos em endereços físicos da rede e vice-versa [TAN97].

Camada de Interface: Responsável pela transferência de dados da máquina origem à máquina destino, fazendo o roteamento e comutação de pacotes. Não-orientado a

conexão, o seu serviço é não confiável e bem simples, sendo responsável pela entrega dos pacotes [TAN1997].

Camada de Endereçamento: Para rotear os datagramas, os gateways analisam os indereços IP no Header do datagrama. Endereço IP é uma palavra de 32 bits, estruturado em classes, que identifica a rede e a estação na rede [TAN1997].

Camada de Aplicação: As aplicações são implementadas de forma isolada, não existindo um padrão de como deva ser estruturada. Elas trocam dados utilizando a camada de transporte (TCP ou UDP) através de chamadas personalizadas, as API's [TAN1997].

3.2 PROTOCOLOS DE BAIXA LATÊNCIA

Vários projetos de pesquisa durante os anos 90 lideravam o desenvolvimento de protocolos de baixa latência que evitariam as intervenções do S.O enquanto outros providenciavam serviços de mensagens a nível de usuários através de redes de alta velocidade. Os protocolos de baixa latência desenvolvidos durante os anos 90 incluía mensagens ativas, mensagens rápidas, o VMMC (Comunicação com Mapeamento Virtual de Memória), U-NET e Interfaces Básicas de Paralelismo (BIP), entre outros[PFI1998].

3.2.1 ACTIVE MESSAGES - MENSAGENS ATIVAS

Mensagens ativas (Active Messages) é uma biblioteca de comunicação de baixa latência projetada pelo Berkeley Network e Workstation. Mensagens curtas e AM são síncronas e são baseadas no conceito de um protocolo Request-Reply. O envio a nível de aplicação do usuário constrói uma mensagem na memória do próprio usuário. Para transferir os dados, o destinatário do processo aloca o buffer receptor, igualmente na memória do usuário e no lado destinatário e solicita uma requisição para o remetente. O remetente responde copiando a mensagem para o buffer do usuário e a envia diretamente para a rede. Não é executada num buffer no sistema de memória. A rede transfere a mensagem para o destinatário, indo a mesma da rede para o buffer destinatário na memória de uso. Esse processo necessita que a memória virtual do

usuário e também os lados remetente e destinatário sejam anexados para um endereço na memória física pois essa não será paginada fora durante a operação na rede.

Entretanto, desde que a ligação do usuário ao buffer de memória esta estabelecida, não é necessário a intervenção do Sistema Operacional para a mensagem ser enviada. Esse protocolo é também chamado de um protocolo "cópia-zero", pois não há cópias da memória do usuário para o sistema de memória em uso.

3.2.2 MENSAGENS RÁPIDAS (Fast Messages)

Fast Messages foi desenvolvido na Universidade de Illinois e é um protocolo similar ao "Active Messages" ou Mensagens Ativas. Na verdade são extensões das Active Messages com impressionantes garantias de robustez nas comunicações básicas. Em particular, Fast Massages garantem que todas as mensagens cheguem seguras e em ordem, até mesmo se o hardware da rede cair. Para isso ser possível, é necessário um controle de fluxo. Se o controle de fluxo não for utilizado para assegurar que o envio rápido não pode ignorar um receptor lento, pode haver então perda de mensagens.

O controle de fluxo é implementado no Fast Massages como um crédito do sistema para que essas mensagens fiquem alocadas na memória do computador Host. Em geral, o controle de fluxos é para prevenir perda de mensagens. Essa é uma complicação que aflige todos os protocolos que requisitam o sistema para mapear a memória principal no computador Host, desde que qualquer um novo buffer possa ser grampeado ou um já grampeado deve de ser esvaziado antes de cada nova chegada de mensagem.

3.2.3 VMMC

O VMMC ou Sistema de comunicações por mapeamento virtual de memória, depois extendido para VMMC-2. É um protocolo de baixa latência criado para o projeto SHRIMP em Princetom. Um objetivo do VMMC é visualizar mensagens como leituras e escritas na memória virtual do sistema, a nível de usuário.

VMMC trabalha pelo mapeamento de páginas de memória virtual do usuário, para a memória física, adicionando uma correspondência entre páginas enviadas e recebidas. VMMC também usa hardware especialmente projetado que permite a

interfaces de redes inserir escritas na memória do Host local e que essas escritas sejam automaticamente atualizadas na memória do Host remoto. Várias otimizações dessas escritas foram desenvolvidas para reduzir o número total de escritas, tráfego na rede e sobrecarga na performance da aplicação. VMMC é um exemplo de um paradigma que sabe como estão distribuídos a memória compartilhada (DSM). Nos sistemas de memória DSM e fisicamente distribuídas entre os nodos em um sistema, mais processos em uma aplicação podem visualizar locações de memória "compartilhada" como idêntica e executar leituras e escritas para as locações.

3.2.4 U-Net

A interface de rede U-Net é uma arquitetura desenvolvida pela Universidade de Cornell e também possibilita "cópia-zero" de mensagens quando possível. U-Net adiciona o conceito de uma interface virtual de memória para cada conexão em uma aplicação usuária. Somente quando um espaço de endereçamento da memória virtual esta mapeado para a memória física real demandada, cada ponto final da comunicação da aplicação é visualizado como uma interface de rede virtual mapeada para um conjunto real de buffers da rede em filas sob demanda.

A vantagem desta arquitetura é que somente o mapeamento é definido, cada interface ativa tem acesso direto para a rede sem intervenção do Sistema Operacional. Como resultado, essa comunicação pode acontecer com uma latência muito baixa.

3.2.5 BIP (Basic Interface of Parallelism)

BIP é um protocolo de baixa latência que foi desenvolvido na Universidade de Lyon. Ele é projetado como uma camada de mensagem de baixo nível. Pode ser construído sob cada camada de alto nível como uma MPI. Programadores podem usar MPI sobre BIP para programação de aplicações paralelas. A interface inicial do BIP consiste no bloqueio e desbloqueio de chamadas. Versões posteriores (BIP-SMP) providenciaram multiplexação entre a rede e a memória compartilhada sobre uma simples API para uso em CLUSTER de multiprocessamento simétrico. Atualmente, BIP-SMP é suportado apenas pelo sistema operacional Linux.

BIP realiza baixa latência e alta taxa de banda pelo uso de diferentes protocolos para vários tamanhos de mensagens e por providenciar copia-zero ou cópia simples de memória para uso de dados. Para simplificar o projeto e ficar com baixo overhead, BIP garante a entrega de mensagens em ordem, embora alguns controles de fluxo escoam para pequenas mensagens e são passadas para um nível se software mais alto.

3.3 PADRÕES DA COMUNICAÇÃO CLUSTER

Em 1997, pesquisas em protocolos de baixa latência foram suficientemente desenvolvidas até estabelecer um novo padrão para comunicação de baixa latência, o VIA ou "Arquitetura de Interface Virtual".

Durante um período similar de tempo a indústria trabalhou na pesquisa de padrões para compartilhar subsistemas de armazenamento. A combinação de esforços entre pesquisadores resultou no padrão Infiniband. Os padrões VIA e Infiniband vão dominar as arquiteturas de rede nos próximos anos [PFI1998].

Um dos principais aspectos que destacam o Infiniband é a não utilização do barramento PCI para conectar periféricos remotos no computador. Com esse propósito, podemos inclusive considerar que o PCI esta no caminho da obsolecência.

3.3.1 VIA

O VIA é um padrão de comunicação que combina muitos dos melhores implementadores e vários projetos acadêmicos. Um consórcio entre Universidades e parceiros industriais, incluindo Intel, Compaq, Microsoft, desenvolveram o padrão. A versão 1.1 do VIA inclui suporte a hardware heterogeneo e foi testado durante o ano de 2001. Igual a U-Net, o VIA é baseado no conceito de interface virtual de rede. Antes, uma mensagem podia ser enviado no VIA, e os Buffers remetente e destinatário são alocados e fixados na memória física.

Chamadas ao sistema não são necessárias após os Buffers e as estruturas de dados serem alocados. Uma operação de envio ou recebimento em uma aplicação do usuário consiste na postagem de um descritor para uma fila. A aplicação pode escolher esperar por uma confirmação até que a operação seja completada ou pode continuar processando no host enquanto uma mensagem é processada.

Vários integradores (comerciantes) de hardware e alguns desenvolvedores independentes tem desenvolvido soluções VIA para vários produtos de rede. As implementações VIA podem ser nativas e/ou emuladas [PFI1998].

3.4 HARDWARE DE REDES

As unidades de execução e fluxo de dados são aspectos importantes para se conseguir um bom desempenho nos sistemas computacionais. Porém, o hardware que compõe as conexões entre os computadores é de fundamental importância para que o fluxo de mensagens trafegue de modo eficiente. O mercado atual oferece diversas soluções que podem proporcionar uma diferença muito grande na performance dos sistemas distribuídos bem como gerar um impacto financeiro sobre os custos de implementação.

3.4.1 PONTES

Esses dispositivos operam na camada de interface da rede do modelo OSI e podem ser utilizados para interconectar redes de tecnologias diferentes (Token Ring, Ethernet, etc). Elas fornecem isolamento de tráfego por segmentos de rede, apresentando-se como solução para o problema de sobrecarga de tráfego em redes locais.

Podemos dividir as pontes em dois tipos:

- Pontes transparentes – Operam em modo promíscuo, ou seja, escutam todos os quadros que são transmitidos nas redes às quais estão conectadas.

- Pontes de roteamento na origem – A estação de origem escolhe o caminho que o quadro ira seguir e inclui a informação de roteamento no cabeçalho do quadro.

A informação de roteamento é construído da seguinte forma: cada Lan e cada porta possuem um identificador único nos seus contextos respectivos, o primeiro bit de endereço de origem dos quadros cujos destino não esta na mesma rede da estação de origem é igual a 1.

3.4.2. ROTEADORES

Trabalham na camada de rede do modelo OSI e desconhecem a posição exata de cada nó, conhecendo só os endereços da sub-rede, não sendo esses transparentes para as pontes pois demandam muita configuração e gerenciamento [TOR1999].

O esquema de endereçamento utilizado pelos roteadores permite que os administradores segmentem a rede em diversas sub-redes, de forma que a arquitetura admitirá várias topologias distintas.

As tabelas de roteamento podem ser:

- Fixas ou estáticas: não são alteradas, e uma vez criadas não sofrem atualizações.
- Dinâmicas: são periodicamente atualizadas.
    - Centralizado – Elemento central
    - Isolado – Apenas informações locais
    - Distribuído – Todos trocam informações
    - Roteamento Hierárquico – Redes divididas em regiões

Também podemos relatar os Multicast que normalmente são separados dos roteadores de "produção" da rede que, uma vez que muitos desses não suportam o IP multicast.

3.4.3 HUB

São equipamentos considerados concentradores de fiação cuja topologia física nem sempre corresponde a topologia lógica. Isoladamente ele não pode ser considerado um equipamento de interconexão de redes; ele só passa a gozar desse "status" quando tem sua função associada a outros equipamentos, como repetidores, por exemplo [TOR1999].

O uso de Hub's torna fácil o isolamento de problemas, bem como facilita enormemente a inserção de novas estações em uma LAN.

3.4.4 SWITCH

É um HUB com funções de ponte e roteador que possui um hardware especial que lhe confere baixo custo e alta eficiência. É uma das opções mais procuradas para responder às crescentes demandas das atuais aplicações em redes [TOR1999].

Também está disponível no mercado de periféricos os Switch USB que conferem um desempenho igual ou superior aos tradicionais do meio Ethernet, podendo alcançar, teoricamente, uma taxa de 200 Mbps.

3.4.5 CABEAMENTOS

Os cabos talvez representem 50% do fracasso ou do sucesso da instalação de uma rede. Muito dos problemas encontrados nas redes são identificados como causados pela má instalação ou montagem dos cabos. Um cabo bem feito contará pontos a seu favor no restante da rede e em caso de dúvidas com algum cabo o melhor é não utiliza-lo [TOR1999].

Os projetos de redes de computadores levam em conta o tipo de cabo que vai ser utilizado na interligação dos nós das mesmas. As tecnologias disponíveis apresentam uma série de modelos de cabos, que diferenciam-se entre si quanto ao material utilizado na sua fabricação (cobre, fibra...), largura máxima de banda (medida em Mbps por segundo), tolerância a ruídos (blindados ou não), distância entre os pontos de conexão, etc.

No caso de sistemas de processamento paralelo e distribuído, o tipo de cabo utilizado é fundamental para se atingir uma velocidade de tráfego alta, com o máximo de eficiência. Abaixo estão relacionados os tipos de cabos mais utilizados e suas características principais de funcionamento.

A tabela demonstra os tipos de cabeamentos mais utilizados e as sua características técnicas baseadas em distância máxima entre pontos e taxa de transferência alcançada. Os valores descritos fazem referência a uma estrutura de cabos estruturada, num cenário adequado ao seu funcionamento.

	10 BASE –T	100 BASE - X	Gigabit Ethernet
Taxa em Mbps	10 Mbps	100 Mbps	1000 Mbps
UTP 5 (m)	100	100	25-100
Cabo coaxial	500	100	25-100
Fibra multimodo	2 Km	400 m (half-duplex) 2 Km (Full-duplex)	500 m
Fibra monomodo	25 Km	20 Km	2 Km

TABELA 1 – MEIOS DE TRANSMISSÃO E DISTÂNCIAS MÁXIMAS SUPORTADAS

3.4.5.1 CABOS COAXIAIS

O primeiro tipo de cabeamento que surgiu no mercado foi o cabo coaxial. Há alguns anos, esse cabo era o que havia de mais avançado, sendo que a troca de dados entre dois computadores era coisa do futuro. Até hoje existem vários tipos de cabos coaxiais, cada um com suas características específicas. Alguns são melhores para transmissão em alta freqüência, outros têm atenuação mais baixa, e outros são imunes a ruídos e interferências. Os cabos coaxiais de alta qualidade não são maleáveis e são difíceis de instalar e os cabos de baixa qualidade podem ser inadequados para trafegar dados em alta velocidade e longas distâncias[TOR1999].

Ao contrário do cabo de par trançado, o coaxial mantém uma capacidade constante e baixa, independente do seu comprimento, evitando assim vários problemas técnicos. Devido a isso, ele oferece velocidade da ordem de Megabits/Seg (Mbps), não sendo necessário a regeneração do sinal, sem distorção ou eco, propriedade que já revela alta tecnologia. O cabo coaxial pode ser usado em ligações ponto a ponto ou multiponto. A ligação do cabo coaxial causa reflexão devido a impedância não infinita do conector. A colocação destes conectores, em ligação multiponto, deve ser controlada de forma a garantir que as reflexões não desapareçam em fase de um valor significativo. Uma dica interessante: em uma rede coaxial tipo BUS - também conhecida pelo nome

de rede coaxial varal , o cabo deve ser casado em seus extremos de forma a impedir reflexões [TOR1999].

A maioria dos sistemas de transmissão de banda base utilizam cabos de impedância com características de 50 Ohm, geralmente utilizados nas TVs a cabo e em redes de banda larga. Isso se deve ao fato de a transmissão em banda base sofrer menos reflexões, devido às capacitâncias introduzidas nas ligações ao cabo de 50 Ohm. Os cabos coaxiais possuem uma maior imunidade a ruídos eletromagnéticos de baixa freqüência e, por isso, eram o meio de transmissão mais usado em redes locais

3.4.5.2 CABOS PAR-TRANÇADO

Com o passar do tempo, surgiu o cabeamento de par trançado. Esse tipo de cabo tornou-se muito usado devido a falta de flexibilidade de outros cabos e por causa da necessidade de se ter um meio físico que conseguisse uma taxa de transmissão alta e mais rápida. Os cabos de par trançado possuem dois ou mais fios entrelaçados em forma de espiral e, por isso, reduzem o ruído e mantém constante as propriedades elétricas do meio, em todo o seu comprimento

A desvantagem deste tipo de cabo, que pode ter transmissão tanto analógica quanto digital, é sua suscetibilidade às interferências a ruídos (eletromagnéticos e radio freqüência). Esses efeitos podem, entretanto, ser minimizados com blindagem adequada. Vale destacar que várias empresas já perceberam que, em sistemas de baixa freqüência, a imunidade a ruídos é tão boa quanto a do cabo coaxial

O cabo de par trançado é o meio de transmissão de menor custo por comprimento no mercado. A ligação de nós ao cabo é também extremamente simples e de baixo custo. Esse cabo se adapta muito bem às redes com topologia em estrela, onde as taxas de dados mais elevadas permitidas por ele e pela fibra óptica ultrapassam, e muito, a capacidade das chaves disponíveis com a tecnologia atual. Hoje em dia, o par trançado também está sendo usado com sucesso em conjunto com sistemas ATM para viabilizar o tráfego de dados a uma velocidade alta igual a 155 Mbps [TOR1999]

3.4.5.3 CABOS DE FIBRA ÓTICA

Quando se fala em tecnologia de ponta, o que existe de mais moderno são os cabos de fibra óptica. A transmissão de dados por fibra óptica é realizada pelo envio de um sinal de luz codificado, dentro do domínio de freqüência do infravermelho a uma velocidade de 10 a 15 MHz. O cabo óptico consiste de um filamento de sílica e de plástico, onde é feita a transmissão da luz. As fontes de transmissão de luz podem ser diodos emissores de luz (LED) ou lasers semicondutores. O cabo óptico com transmissão de raio laser é o mais eficiente em potência devido a sua espessura reduzida. Já os cabos com diodos emissores de luz são muito baratos, além de serem mais adaptáveis à temperatura ambiente e de terem um ciclo de vida maior que o do laser

Apesar de serem mais caros, os cabos de fibra óptica não sofrem interferências com ruídos eletromagnéticos e com radio freqüências e permitem uma total isolamento entre transmissor e receptor. Portanto, quem deseja ter uma rede segura, preservar dados de qualquer tipo de ruído e ter velocidade na transmissão de dados, os cabos de fibra óptica são a melhor opção do mercado

O cabo de fibra óptica pode ser utilizado tanto em ligações ponto-a-ponto quanto em ligações multiponto. A exemplo do cabo de par trançado, a fibra óptica também está sendo muito usada em conjunto com sistemas ATM, que transmitem os dados em alta velocidade. O tipo de cabeamento mais usado em ambientes internos (LANs) é o de par-trançado, enquanto o de fibra óptica é o mais usado em ambientes externos[TOR1999].

3.4.6 ADAPTADORAS DE REDES

As adaptadoras de redes (ou placas de redes) são os dispositivos que fazem o interfaceamento entre máquinas, ou seja, por elas é que os dados são recebidos ou enviados junto ao computador.

Elas se diferenciam pela qualidade dos componentes utilizados na sua confecção, taxas de transferências de dados, tipo de barramento de conexão (PCI, ISA,...) e padrões de conectores (RJ 45, etc). [TOR1999]. Podem ser encontradas em várias faixas de custos e de desempenho, conforme especificação de seu fabricante.

3.4.6.1 ETHERNET

As adaptadoras Ethernet são as mais simples e de baixo custo no mercado, possibilitando uma taxa de transmissão de 10 Mbps, o que satisfaz as redes de pequeno porte, onde estão conectados no máximo 10 computadores. Elas podem suprir com tranqüilidade um ambiente SOHO (Small Office Home Office), locais esses que exigem soluções de baixo custo [TOR1999].

O tipo de barramento de conexão suportada por essas adaptadoras varia do ISA até o PCI (ver tópicos). Isso significa que qualquer computador pessoal tem a capacidade de suportar uma adaptadora desse padrão, independente do modelo ou fabricante.

3.4.6.2 FAST-ETHERNET

As placas Fast-Ethernet são uma evolução das adaptadoras Ehternet, com o diferencial de permitir uma maior largura de banda no tráfego de dados, nesse caso em 100 Mbps, agilizando muito o transporte de pacotes na rede. Ë uma solução que atualmente possui um custo bastante baixo, podendo ser aplicado sem problemas as pequenas soluções de redes.

Existe uma grande variedade de fabricantes e modelos, e a qualidade dessas adaptadoras depende dos componentes integrados a elas. Estão disponíveis para os barramentos PCI, tanto no de 33 Mhz e 32 bits, quanto no de 66 Mhz e 32 Bits.

3.4.6.3 GIGABIT ETHERNET

É uma categoria especial entre as adaptadoras de redes, cujo maior diferencial é permitir uma largura de banda de até 1 Gbits/s. O alto desempenho vem aliado a um custo maior, o que a coloca como solução ideal em ambientes de trabalho que necessitam um alto tráfego de dados, incluindo áudio e vídeo com um mínimo de qualidade, suportando Videoconferência sem maiores problemas quando em canais dedicados.

As adaptadoras dessa categoria se diferenciam das demais principalmente na arquitetura que contempla processamento e memórias otimizadas, podendo ser interligadas por cabos de cobre ou fibra ótica.

3.5 BARRAMENTO

Em um ambiente computacional, diversos sub-sistemas precisam ter interfaces uns com os outros. Por exemplo, a memória e o processador precisam se comunicar, assim como o processador e os dispositivos de entrada e saída. Essa comunicação é feita em geral por um barramento. Ele nada mais é do que um link de comunicação compartilhado que usa um conjunto de fios para a conexão dos vários subsistemas [PAT2000].

Vantagens: Versatilidade e o baixo custo;

Desvantagem: Cria-se um gargalo na comunicação, em geral limitada a throughput máximo das operações de entrada e saída.

A velocidade do barramento é limitada por fatores físicos; o comprimento do barramento e o número de dispositivos nele conectados. Ele é composto em geral por um conjunto de linhas de dados e uma de linhas de controle.

Linhas de Dados: Transportam informações da fonte para o destino. A informação pode ser composta de dados, comandos complexos ou endereços.

Linhas de Controle: Usadas para sinalizar solicitações (acks), além de indicar o tipo de informação que esta na linha de dados.

3.5.1 ARBITRAGEM DOS BARRAMENTOS

É a decisão sobre qual dispositivo vai obter controle do barramento. Existe uma grande variedade de esquemas para arbitragem [PAT2000]:

Daisy Chain: A linha com a informação da garantia de uso do barramento esta ligado a todos os dispositivos conectados ao barramento, à partir do dispositivo de mais alta prioridade até o de mais baixa prioridade (as prioridades são determinadas pela posição do dispositivo no barramento). A vantagem desse modelo é a simplicidade, mas em contrapartida é que nem todos os dispositivos tem acesso ao barramento.

Arbitragem centralizada com requisição em paralelo: Esse esquema usa diversas linhas para requisição de acesso, sendo que os dispositivos podem acessar o

barramento de uma forma independente um do outro. Um árbitro centralizado escolhe um entre os dispositivos que solicitam o uso do barramento, e informa então que esse dispositivo então, que ele é o mestre do barramento. A desvantagem é o gargalo para o uso do barramento, modelo esse adotado pelo padrão PCI.

Arbitragem distribuída com acesso por auto-seleção: Cada dispositivo coloca no barramento o código que o identifica. A partir do exame da informação constante no barramento, os próprios dispositivos podem determinar qual deles é o de mais alta prioridade que esta requisitando o barramento naquele instante. Ex.: NuBus do Apple Mac II.

Arbitragem distribuída com acesso por controle de colisões: O acesso é independente. Modelo adotado pelas redes padrão Ethernet [TOR1999].

3.5.2 TIPOS DE BARRAMENTOS

As atuais tecnologias de barramento são orientadas para dois tipos distintos: os paralelos e os seriais.

Paralelos: É um modelo onde os dados são enviados simultaneamente por diversas trilhas. Os barramentos paralelos são os mais utilizados atualmente, por serem mais rápidos e exigirem um protocolo de comunicação mais simples. É o caso do barramento PCI.

Seriais: É um modelo que possui somente uma trilha dedicada aos dados, sendo portanto de implementação física mais simples mas que obriga a implementação de protocolos mais complexos.

Outra característica que define a agilidade do barramento é saber se ele é síncrono ou assíncrono.

Síncrono: É qualificado como Síncrono quando o clock é utilizado como sinalizador (ou maestro), orientando o tráfego de pacotes de dados ou reservando o barramento para uma nova transmissão.

Assíncrono: É qualificado como Assíncrono quando não possui um sinalizador (ou maestro) para orientar os dispositivos conectados ao barramento ou as transmissões [TOR1999].

Tecnologias	Throughput Máximo Megabits	Throughput Máximo Megabytes
Apple Desktop Bus	0.01 Mbps ou 10 Kbps	0.0013 MB/s
Porta Serial	0.23 Mbps ou 230 Kbps	0.029 MB/s
USB 1.0	1.5 Mbps	0.19 MB/s
10Base-T	10 Mbps	1.25 MB/s
USB 1.1	12 Mbps	1.5 MB/s
Fast SCSI	80 Mbps	10 MB/s
100Base-T	100 Mbps	12.5 MB/s
Ultra – SCSI	160 Mbps	20 MB/s
Wide Ultra – SCSI	320 Mbps	40 MB/s
Ultra 2 – SCSI	320 Mbps	40 MB/s
FireWire	400 Mbps	50 MB/s
USB 2.0	480 Mbps	60 MB/s
Wide Ultra 2 – SCSI	640 Mbps	80 MB/s
FireWire 2	800 Mbps	100 MB/s
Ultra 3 - SCSI	1280 Mbps	160 MB/s

TABELA 2. COMPARATIVO DE PERFORMANCE DE TECNOLOGIA DE COMUNICAÇÃO

3.5.3 ISA

O barramento ISA, apesar das limitações técnicas de sua especificação (16 bits e 8MHz de clock), permite que se projetem placas de expansão simples, baratas e que oferecem desempenhos suficientes para determinados periféricos, como placas fax-modem, placas de som e portas de comunicação serial e paralela.

Frente ao modismo plug and play, surgiu a necessidade de estabelecer-se um modelo que permitisse a configuração dinâmica dos recursos de hardware para as placas ISA. Vale observar que o arranjo de sinais elétricos do barramento que é usado pelas placas ISA plug and play é o mesmo que o especificado pela IBM, em 1984, por ocasião do projeto do PC AT.

3.5.4 PCI

Motivado pelas limitações técnicas do barramento ISA (8MHz, 16 bits), a Intel, em 1992, introduziu a especificação de barramento PCI (Peripheral Component

Interconnect), que permite a comunicação de palavras de 32 ou 64 bits, a 33MHz. Mais que uma nova especificação de barramentos para PCs, o PCI possui a característica de universalidade, ou seja, pode ser aproveitado por qualquer processador em qualquer arquitetura de máquina. Isto parece ser uma dualidade, pois, apesar de universal, o PCI também tem característica de barramento local.

Outra inovação do barramento PCI é expandir a limitação de alguns carregamentos elétricos através de pontes PCI-PCI. Uma ponte deste tipo cria um barramento secundário isolado eletricamente do barramento raiz, permitindo-se a utilização de mais placas de expansão.

Esta característica consagrou o PCI como barramento nas arquiteturas de servidores de grande desempenho. Além da ponte PCI-PCI, foi especificada a ponte PCI-ISA, de forma a permitir o aproveitamento de placas ISA na configuração da máquina.

Características	**PCI**
Tipo de Barramento	BackPlane
Comprimento do Barramento	32 a 64 bits
Multiplexação de endereços e dados ?	Multiplexados
N º Mestres do Barramento	Vários
Arbitragem	Centralizado com requisição em paralelo
Clock	Síncrono 33 a 66 MHz
Banda passante / Pico	133 a 528 MB/s
Banda passante / Prática	80 MB/s
N º Máximo de dispositivos conectados	1024 – 32 por segmento
Comprimento Máximo do Barramento	0,5 metros
Nome padrão	PCI

TABELA 3 - CARACTERÍSTICAS TÉCNICAS DO BARRAMENTO PCI

Ao contrário do barramento ISA, podem existir múltiplos barramentos PCI numa configuraçào de máquina (a vantagem disto é descongestionar o fluxo de dados onde está conectada a CPU principal).

Tais barramentos são hierarquizados da seguinte forma: existe um barramento raiz e os outros barramentos são interligados eletricamente por pontes PCI-PCI.

Freqüência do Barramento Local	Freqüência do Barramento PCI	Taxa de Transferência
50 Mhz	25 Mhz	100 MB/s
55 Mhz	27,5 Mhz	110 MB/s
60 Mhz	30 Mhz	120 MB/s
66 Mhz	33 Mhz	132 MB/s
75 Mhz	37,5 Mhz	150 MB/s
83 Mhz	41,5 Mhz	166 MB/s
100 Mhz	33,3 Mhz	133,2 MB/s

TABELA 4 – FREQÜÊNCIAS DO BARRAMENTO PCI EM RELAÇÃO AO USO DE PROCESSADORES.

Barramento de Dados	Freqüência do barramento PCI	Taxa de transferência máxima
32 bits	33 Mhz	132 MB/s
32 bits	66 Mhz	264 MB/s
64 bits	33 Mhz	264 MB/s
64 bits	66 Mhz	528 MB/s

TABELA 5 – TAXAS DE TRANSFERÊNCIA DE DADOS VIA FREQÜÊNCIA DO BARRAMENTO PCI

Porém, esse arranjo de pontes é que limitam bastante a taxa de transferência do barramento, já que máquinas de baixo custo se utilizam do padrão PCI 32 bits, trabalhando a uma freqüência de 33 MHz, e pior, sendo compartilhados com os outros dispositivos instalados no computador (placas de vídeo, placas de som, etc). As opções de computadores que possuem conexões PCI 32 bits com uma freqüência de 66 MHz, o que garante um pouco mais de peformance, não são oferecidos como opção em PC comerciais de baixo custo, ficando restritos a equipamentos de nível e custo superior.

3.5.5 FIREWIRE

Se o padrão USB é ideal para periféricos de baixa e média velocidade, o que fazer com aqueles que exigem mais rapidez? Para suprir essa necessidade, existe o FireWire, algo como "fio de fogo". Também conhecido como IEEE 1394, ele atinge velocidades de 100, 200 e 400 Mbps, contra o máximo de 12 Mbps do USB versão 1.0/1.1, e permite que os dispositivos conectados se comuniquem entre si, sem a intervenção do computador, além de que cada porta Firewire permite a conexão de até 63 periféricos simultaneamente.

Mais recentemente foi incorporado a segunda versão do barramento FireWire, disponibilizando uma taxa de transferência máxima de 800 Mbps segundo os fabricantes. Se essa taxa nominal é de fato atingida, somente os experimentadores

poderão confirmar, mas não deixa de ser mais uma opção interessante para ligar periféricos ao computador.

3.5.6 USB (Universal Serial Bus)

É uma interface plug and play entre o computador e dispositivos (como joystick, MP3 Player, teclados, telefones, scanner, e impressoras). Com o USB um novo dispositivo pode ser colocado no computador sem ter que adicionar uma placa adaptadora ou mesmo ter de desligá-lo. Basta conectar o periférico que ele é automaticamente identificado e instalado.

O padrão USB foi desenvolvido pela Compaq, IBM, DEC, Intel, Microsoft, NEC, e Northern Telecom e a tecnologia está disponível sem custos para todas as empresas que desejarem. O USB suporta uma velocidade de transferência de dados de 12 Mbits por segundo (versão 1.1), e 480 Mbits por segundo (versão 2.0), trabalhando a uma freqüência de 48 MHz na versão 1.1 podendo chegar a 480 MHz na 2.0. Esta velocidade é suficiente para uma enorme variedade de dispositivos. Pode-se conectar cerca de 127 periféricos USB em um computador! [PAL2002].

Existem três formas de se implementar um Host USB nas Mother-Board comercializadas atualmente. A primeira é a ligação direta no FSB (Front Side Bus) do computador, ou através de uma ponte PCI-USB e também por um cartão ou placa de expansão conectada no barramento PCI. Obviamente existem diferenças de performance entre os 3 tipos de implementação.

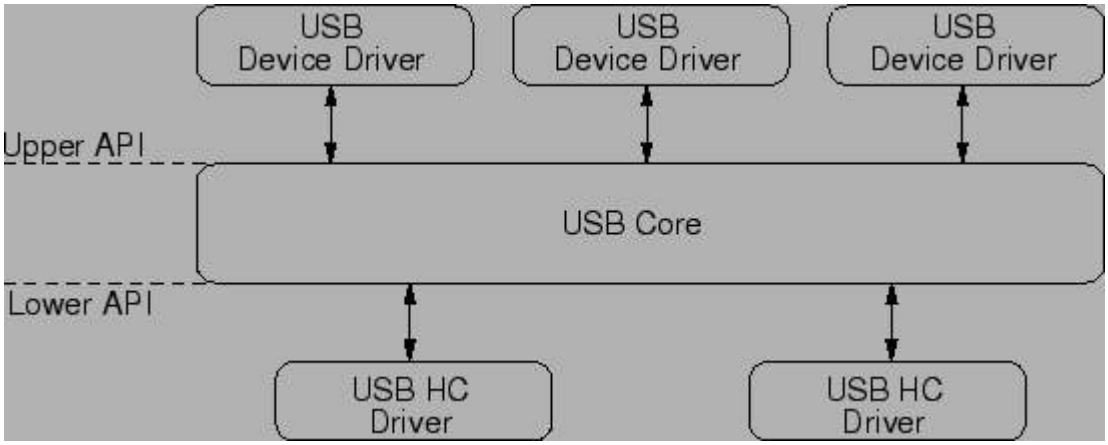

FIGURA 4 – ESQUEMA DE UMA IMPLEMENTAÇÃO USB

4 MULTICOMPUTADOR CRUX

O CRUX é o objeto de estudos desse trabalho, e como tal, podemos descrever algumas características desse Multicomputador permitindo uma compreensão mais clara do seu funcionamento. Não é objeto de estudos as plataformas PC em que ele pode ser implementado, mas sim explorar meios para otimizar o desempenho e comunicação entre os nós que fazem parte de sua arquitetura.

O multicomputador CRUX é composto de um conjunto de nós de trabalho (NT) ligados por um comutador de conexões e um barramento compartilhado. O comutador de conexões (Crossbar) é manipulado durante o funcionamento normal da máquina pelo nó de controle (NC), que por sua vez utiliza o barramento de serviço e um canal de configuração para determinar as necessidades do sistema e definir din6amicamente a estrutura da rede de comunicação.

O nó de controle utiliza o barramento de serviço para realizar uma pesquisa seqüencial de modo a determinar os pedidos dos nós de trabalho. Esta pesquisa é feita através do questionamento de envio de comando a um nó inquirido. Caso este nó não deseje nenhum serviço, o nó de controle questiona o próximo nó de trabalho. Entretanto, se esse nó questionado deseja o serviço, ele então envia uma requisição ao NC. Após atender cada pedido, o nó de controle volta ao questionamento seqüencial dos nós de trabalho. Se o pedido for de conexão com outro nó de trabalho, o NC envia uma nova configuração ao comutador de conexões. Se o nó solicitado já estiver conectado, o pedido é colocado numa fila e o nó de controle prossegue com o questionamento seqüencial.

Os nós de trabalho (NT) são computadores pessoais, com processador, microcódigo, memória própria e dispositivos de E/S, incluindo a interface com o comutador de conexões, com 4 canais físicos de interface com o barramento de serviço.

O nó de controle (NC) é responsável pelo controle do comutador de conexões (crossbar), através de um canal de configuração. Todos os demais nós podem requisitar conexões poto-a-ponto com outro nó de trabalho, comunicado com o nó de controle através do barramento de serviço. O que distingue o NC dos NT é a capacidade de comandar o comutador de conexões. Devido a essa atribuição, ele é dedicado

exclusivamente as tarefas relacionadas com essas funções, e não utiliza o crossbar para se comunicar com outros nós.

O barramento de serviço é o canal de comunicação entre os NT e o NC, e é utilizado apenas para a troca de mensagens curtas, com requisições e respostas pré-estabelecidas. O NC verifica, seqüencialmente, se há algum NT que deseje se comunicar com ele, quando então recebe uma mensagem de requisição do NT interessado. O NC permanece nesse *polling* indefinidamente [COR1996].

De uma forma geral, aplicações de tempo real apresentam requisitos diferentes das aplicações convencionais "não tempo-real". Dentro do ambiente de aplicação da máquina CRUX propõe-se um conjunto mínimo de primitivas de tempo-real, baseadas no padrão POSIX da IEEE.

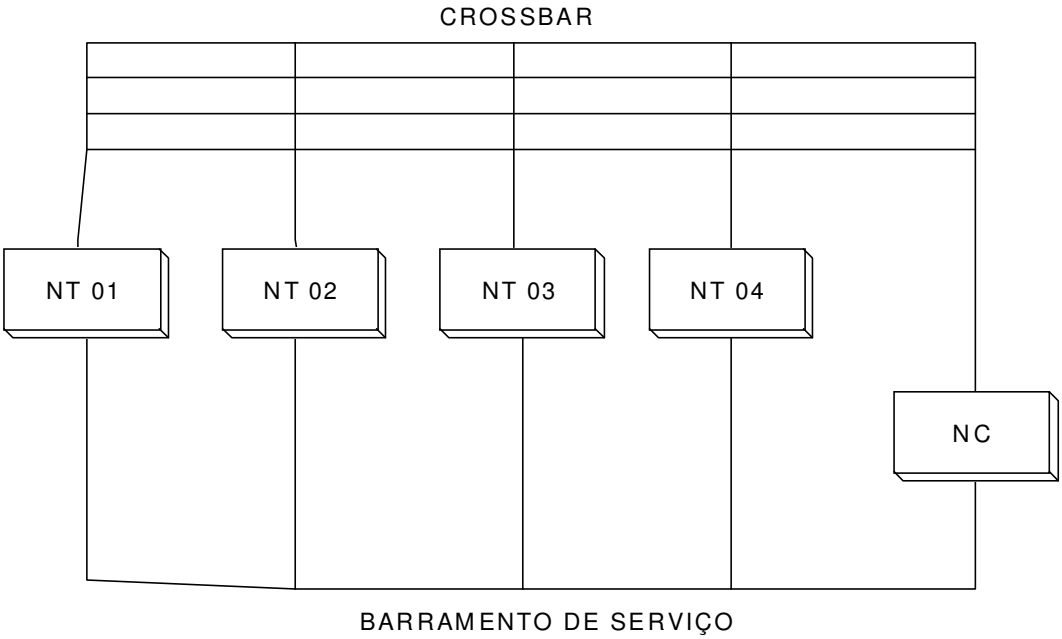

FIGURA 5 – ARQUITETURA DO MULTICOMPUTADOR CRUX

4.1 CRIAÇÃO, GERENCIAMENTO E ESCALONAMENTO DE THREADS

Tendo em vista que na máquina CRUX tem apenas um processo sendo executado por processador, não se faz necessário um controle no escalonamento dos processos, pois estes são alocados em um dado processador no momento de sua criação. No entanto, cada processo pode ser multithread, ou seja, pode ter mais de uma thread de

execução. Isto permite uma maior flexibilidade por parte das aplicações de tempo real. Entretanto, é necessário realizar o gerenciamento dessas threads. A opção foi pela utilização de um número mínimo de primitivas de suporte, de modo a facilitar a sua utilização a nível do núcleo ou a nível de usuário.[COR1996]

4.2 GERÊNCIA DE MEMÓRIA

Uma vez que o CRUX associa a cada processador um único processo, torna-se desnecessário uma gerência de memória virtual. Isso é muito desejável, já que o gerenciamento de memória virtual, como paginação ou swapping, traz consigo uma imprevisibilidade associada, pois o benefício de tornar menor o tempo de acesso não pode ser considerado uma vez que em tempo real temos que considerar o pior caso (que, no caso, seria a informação não estar em cache) [COR1996].

4.3 COMUNICAÇÃO ENTRE PROCESSOS/THREADS

A comunicação entre processos e threads, no ambiente CRUX, é realizada, exclusivamente, por passagem de mensagens. Essa passagem de mensagens pode ser feita utilizando canais de comunicação.

A comunicação em um sistema de tempo real necessita ser, antes de mais nada, previsível, ou seja, é necessário a garantia de que uma determinada mensagem seja recebida/atendida em um tempo máximo (deadline), ou ainda, seja enviada em um determinado instante.

As primitivas utilizadas permitem que seja estabelecida a comunicação por passagem de mensagens, implementando filas de mensagens. Desta forma, é possível a utilização, tanto de mecanismos de canais quanto de mecanismos de caixas postais (se desejável), associando esses mecanismos a uma determinada fila de mensagem [COR1996].

5 REDES USB

Implementar uma rede conectado a interface USB de um computador é uma opção bastante interessante. Porém, as dúvidas de como ele vai se comportar e se o desempenho vai ser bom só podem ser resolvidas avaliando essa tecnologia e as promessas do mercado em relação a ela. A topologia do barramento USB é no formato estrela, através do uso de hub's para expandir o número de dispositivos ligados a ele.

Podemos encontrar os conectores USB nas placas-mãe comercializadas a partir de 1996, porém, ligadas em adaptadoras PCI–USB. Essas adaptadoras ainda podem ser encontradas no comércio de periféricos.

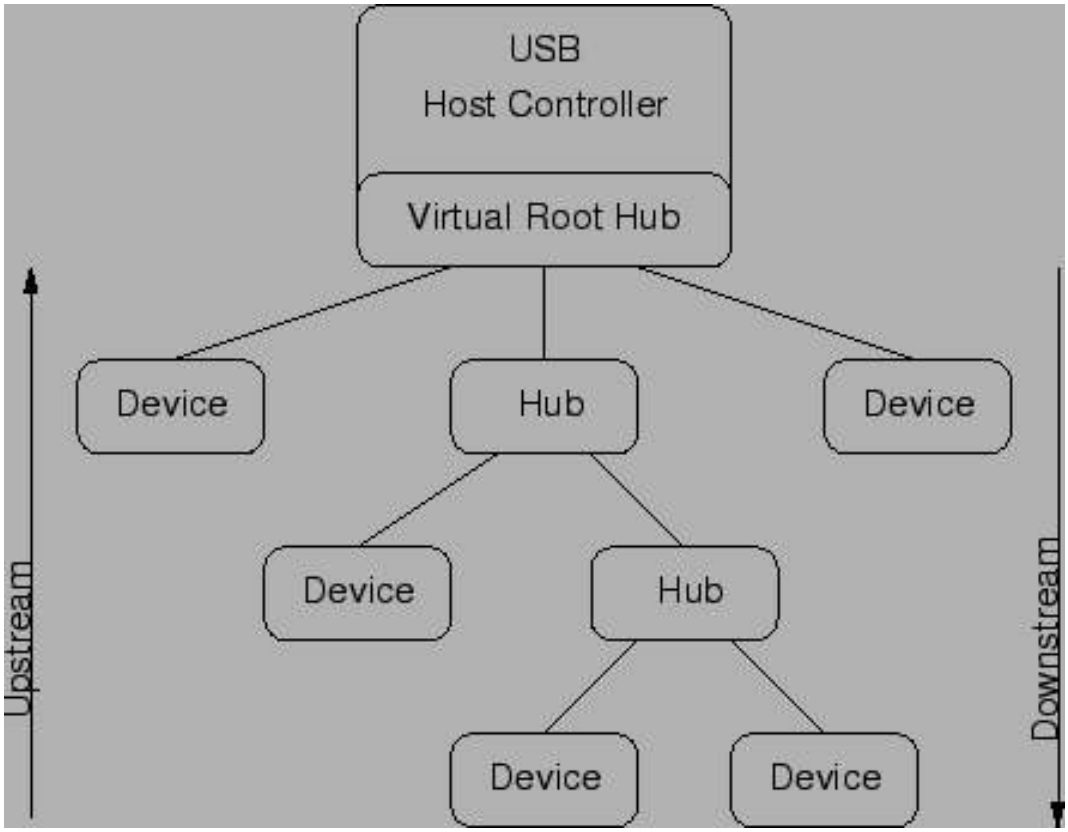

FIGURA 6 – TOPOLOGIA USB

Todos os dispositivos ligados ao host USB ou ao hub são identificados com um número de identificação único. O barramento USB permite conectar um número máximo de até 127 periféricos e com no máximo 16 em cada hub, cada um deles com seu respectivo endpoint.

5.1 USB - CARACTERÍSTICAS TÉCNICAS

O barramento físico do padrão USB é composto de um cabo blindado (para evitar irradiações eletro-magnéticas) com quatro condutores: **Vbus, D+. D-** e **GND**. O sinal Vbus é responsável pelo fornecimento da alimentação elétrica para os dispositivos USB, de forma que esses não necessitem fontes próprias, o que é uma forma de baratear custos de projeto. A troca de informações entre os dispositivos acontece via diferença entre as tensões dos sinais **D+** e **D-**. Os dois terminais de cabo possuem conectores diferentes para os encaixes *upstream* e *downstream* .

PINO	SINAL	COR
1	Vbus (+5 V)	Vermelho
2	D-	Branco
3	D+	Verde
4	GND	Preto

TABELA 6 – PINAGEM DO CABO USB

Os cabos USB possuem uma impedância diferencial de 90 Ù. A posição dos resistores de pull-up muda dependendo de tratar-se de alta ou baixa velocidade. Na especificação 1.0/1.1, um pull-up ligado a D+ indica operação de 12 Mbps, enquanto que um pull-up ligado a D- significa operação em 1,5 Mbps. Quando não existe função conectada ao hub, os resistores de pull-down levam ambos D+ e D- a um valor menor do que uma tensão de limiar para a detecção da presença de um dispositivo. Se essa condição persistir por mais de 2,5 ì s, é caracterizada a desconexão do dispositivo.

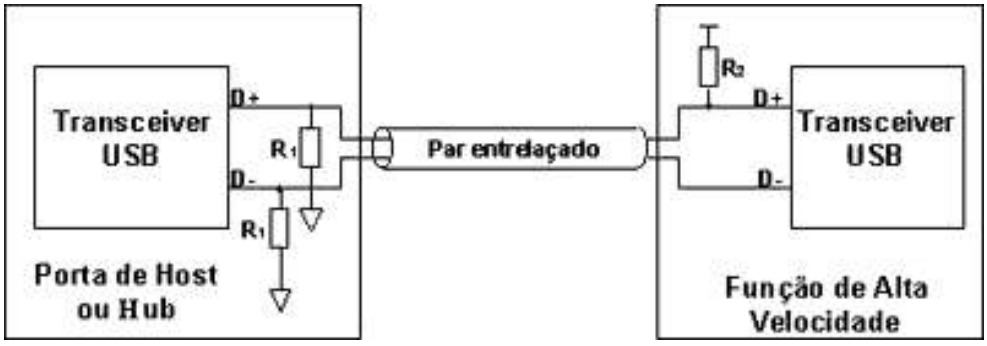

FIGURA 8 – ESQUEMA FÍSICO DOS SINAIS USB

Na especificação 1.0/1.1, uma função pode drenar até 100mA pelo cabo USB, sendo que esse valor pode atingir os 500mA, caso a função tenha alimentação própria. Para permitir uma taxa de 12 Mbps, o tamanho do cabo USB não deve ter mais de 5 m [MEN2002].

5.2 A COMUNICAÇÃO SERIAL

Foi descrito anteriormente que existem dois sinais (D+ e D-) responsáveis pela comunicação serial entre o periférico e o hub. A especificação USB define 3 estados lógicos:

ESTADO	D+	D-
J	ALTO	BAIXO
K	BAIXO	ALTO
SE0	BAIXO	BAIXO

TABELA 7 – ESTADOS LÓGICOS DO USB

Quando não existe atividade no barramento, diz-se que que ele esta ocioso (idle). Neste caso, entrará em estado J, ou seja, D+ estará em nível alto e D- em nível baixo. Já para o caso de não haver dispositivos conectados, os dois resistores Pull-down garantem um estado SE0 no barramento [MEN2002].

A comunicação entre os dois terminais USB é realizada mediante transações, que são constituídas pela transmissão de conjuntos de pacotes. O primeiro pacote é sempre iniciado por um hub (o raiz ou um externo), que assume o papel de iniciador/mestre do barramento. Dependendo do tipo de transação. Os pacotes seguintes podem ser iniciados tanto por um hub, quanto por uma função (periférico).

Um pacote possui pelo menos 3 campos: sincronismo (SYNC), identificação do tipo de pacote (PID) e fim de pacote. O sincronismo é constituído pela transmissão da seqü6encia de estados KJKJKJKK. Assim, o sincronismo, o identificador do tipo de pacote (PID) também é transmitido pelo hub, com a característica de ser totalmente definido pelos quatro primeiros estados (KJJJ, por exemplo). Em seguida, dependendo do tipo de pacote, outros campos de dados (até 1024 bytes) podem ser transmitidos,

sendo que a direção do fluxo de informações também depende do tipo de pacote. Todo pacote é encerrado com a transmissão de dois estados SE0, quando o barramento volta a ficar ocioso.

Caso o hub amostre SE0 por três períodos de relógio (12 MHz) consecutivos, ou seja, por 2,5 ì s, o hub considera que o dispositivo foi desconectado fisicamente.

PID	NOME	TIPO
0101	SOF	TOKEN
1101	SETUP	
1001	IN	
0001	OUT	
0011	DATA0	DATA
1011	DATA1	
0010	ACK	HANDSHAKE
1010	NAK	
1110	STALL	

TABELA 8 – IDENTIFICADORES DE PACOTES USB

SOF – Start of Frame : O pacote SOF é transmitido pelo hub raiz a uma taxa fixa de 1000 pacotes/segundos (um a cada milissegundo)e não tem grandes aplicações práticas. O intervalo entre dois SOFs consecutivos é chamado de quadro, ou frame.

Neste pacote o Hub, após transmitir o PID 0101 (SOF)`transmite também o valor de um contador crescente de SOFs de 11 bits, ou seja, um contador com módulo 2048. O hub transmite em seguida 5 bits redundantes para a checagem de erros de transmissão, usando a codificação tipo CRC (Cyclic Redundancy Check) [MEN2002].

KJKJKJKK	8 bits	11 bits	5 bits	2 x SE0
SYNC	PID	Contador de SOF	CRC 5	Fim

TABELA 9 -FORMATO DO PACOTE SOF

SETUP, IN e OUT: Esses três tokens são transmitidos por hubs e tem a finalidade de selecionar quem será o próximo endereço (dispositivo) e o próximo *endpoint* (sub-dispositivo) que serão os alvos da transação seguinte, possivelmente com pacotes DATA0 ou DATA1. A diferença entre eles é que o token IN indica que o próximo pacote tipo *data* será para a leitura de dados na função, enquanto que o token tipo OUT indica que a transação será para a escrita na função. O token SETUP é semelhante ao token OUT, porém não pode ser rejeitado pela função e é sempre endereçado ao *endpoint0*. O campo que escolhe o o endereço do dispositivo (ADDR) contém 7 bits, contra 4 do campo que seleciona o *enpoint* (ENDP). Novamente, 5 bits redundantes são utilizados para a checagem de erros de transmissão [MEN2002].

KJKJKJKK	8 bits	7 bits	4 bits	5 bits	2 x SE0
SYNC	PID	ADDR	ENDP	CRC 5	Fim

TABELA 10 -FORMATO DO PACOTE IN, OUT E SETUP

DATA1 E DATA0: Os pacotes DATA1 e DATA0 transportam a informação propriamente dita que trafegará entre o Hub e a função. Caso o último pacote de token transmitido tenha sido um OUT ou um SETUP, todo o pacote, incluindo os dados de informações, respeita um fluxo do tipo *downstream* (do Hub para a função). Caso o último pacote de token transmitido tenha sido um IN, o fluxo é do tipo *upstream* (da função para o Hub) [MEN2002].

KJKJKJKK	8 bits	0 a 1023 x 8 bits	16 bits	2 x SE0
SYNC	PID	Dados de informação	CRC 16	Fim

TABELA 11 -FORMATO DO PACOTE DATA0 E DATA1

ACK, NACK e STALL: Eles são compostos exclusivamente pelos campos obrigatórios, ou seja: SYNC, PID e FIM. O fluxo de trasmissão é contrário ao fluxo de dados. Por exemplo, caso o pacote do tipo DATA tenha seguido o fluxo *upstream*, o pacote *handshake* seguinte terá seguido o fluxo *downstream* e vice-versa. Basicamente um ACK serve para informar que o último pacote do tipo DATA ou token foi recebido corretamente. Já o NACK indica que o receptor esta ocupado ou inapto a realizar à comunicação naquele momento. Um STALL indica que o receptor detectou erros de comunicação [MEN2002].

KJKJKJKK	8 bits	2 x SE0
SYNC	PID	Fim

TABELA 12 -FORMATO DO PACOTE DATA0 E DATA1

A transmissão de dados via USB é baseada no envio de pacotes. A transmissão começa quando o Controlador Host envia um pacote (Token Packet) descrevendo o tipo e a direção da transmissão, o endereço do dispositivo USB e o referido número de endpoint.

A transmissão de dados pode ser realizada tanto do Host para o dispositivo quanto em sentido inverso. O dispositivo USB decodifica o campo de endereço, reconhecendo que o pacote lhe é referente. A seguir, a fonte da transmissão envia um pacote de dados (Data Packet) ou indica que não há dados a transferir. O destino responde com um pacote de Handshake (Handshake Packet) indicando se a transferência obteve sucesso [MEM2002].

O USB utiliza três tipos de pacotes: Token, Data e Handshake Packets, mostrados nas figura 11 (a), (b) e (c), respectivamente. Esses pacotes possuem os seguintes campos:

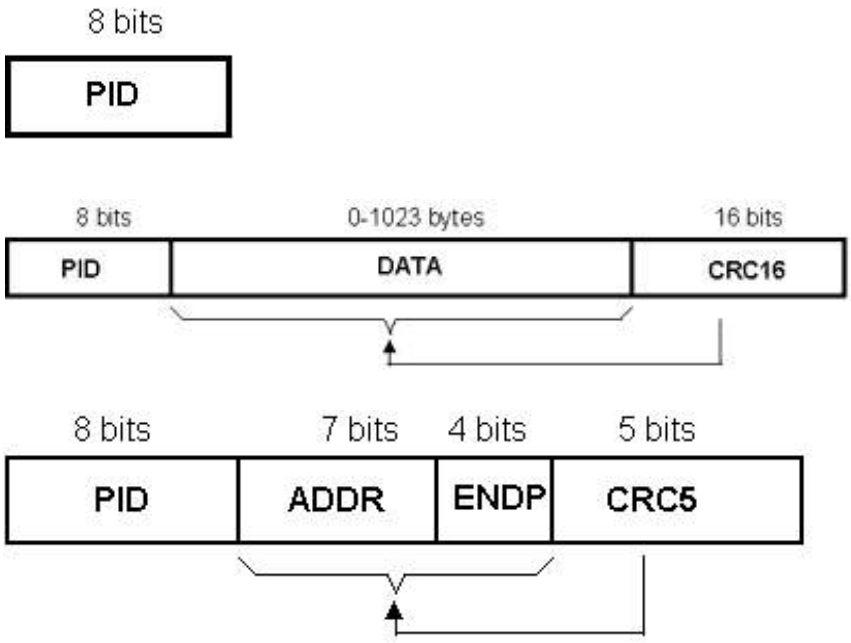

FIGURA 11 – PACOTES USB – CONTROL, DATA E HANDSHAKE

PID (Packet Identifier): composto de oito bits. Os quatro mais significativos identificam e descrevem o pacote e os restantes são bits de verificação para prevenção de erros (check bits). Esses check bits são constituídos pelo complemento um dos quatros bits identificadores;

ADDR (Address): endereço do dispositivo USB envolvido. Composto de 7 bits, limita o número de dispositivos endereçáveis em 127;

ENDP (Endpoint): possui 4 bits que representam o número do endpoint envolvido. Permite maior flexibilidade no endereçamento de funções que necessitem de mais de um subcanal;

CRC (Cyclic Redundancy Checks): bits destinados à detecção de erros na transmissão;

DATA : bits de dados.

Um Token Packet pode identificar a transmissão como sendo de transferência para o Host (IN), de transferência para a função (OUT), de início de frame (SOF) ou de transferência de informações de controle para o endpoint (SETUP). O CRC de um Token Packet possui 5 bits e atua apenas sobre os campos ADDR e ENDP, uma vez que o PID possui seu próprio sistema de prevenção contra erros. Os dados transmitidos via Data Packet devem ter um número inteiro de bytes. O CRC de um Data Packet possui

16 bits e age apenas sobre o campo DATA. O Handshake Packet é constituído apenas de um PID. Esse pacote pode significar que o receptor recebeu os dados livres de erros (ACK), que o receptor não pode receber os dados, que o transmissor não pode transmitir (NAK) ou que o endpoint está em parado (STALL). O USB aceita quatro tipos de transferências diferentes: **Control, Bulk, Interrupt e Isochronous**.

A transferência do tipo **Control** serve para configurar ou transmitir parâmetros de controle a um dispositivo. Inicialmente, em idle, ele recebe um Token de SETUP oriundo do Controlador Host. Em seguida, o Host envia um Data Packet para o endpoint de controle da função. A função envia, então, ao Host um Handshake Packet de reconhecimento (ACK) e entra em idle.

A transferência **Bulk** é utilizada para a transmissão de grande quantidade de dados, como em impressoras ou scanners. Ela garante uma transmissão livre de erros por meio da detecção de erros e de novas retransmissões, se necessário. Caso o Host deseje receber uma grande quantidade de dados, ele envia um Token de IN e a função devolve um Data Packet. Se houver algum problema, a função envia um STALL ou NAK e entra em idle. Ao final, o Host devolve um ACK. Se, em vez de receber, o Host desejar enviar dados, ele manda um Token de OUT em vez de IN.

A transmissão do tipo **Interrupt** é requisitada pelo Host e consiste numa transferência de pequena quantidade de dados. Os dados podem representar a notificação de algum evento, como os de um mouse ou caneta ótica.

A transferência tipo **Isochronous** permite o tráfego de dados que são criados, enviados e recebidos continuamente em tempo real. Nessa situação não há handshake, devido à própria continuidade com que os dados são transmitidos. Caso contrário, haveria atraso e a transmissão em tempo real seria comprometida. Os pacotes do tipo *data* são DATA0 e podem ter até 1023 bytes. Todas as especificações técnicas do padrão USB estão rigorosamente estabelecidas na Universal Serial Bus Specification Revision 1.0 [AND2000].

5.2.1 CODIFICAÇÃO CRC

O CRC (Cyclic Redundancy Check) é um método de detecção de erros de alta eficiência. Ele é gerado por restos sucessivos de uma divisão módulo 2 usando um divisor previamente combinado. A divisão módulo 2 é igual a uma divisão comum onde a subtração normal é substituída pela subtração módulo 2. A preferência por esse tipo de operação vem de sua facilidade de implementação em hardware, pois bastam portas ou-exclusivo (XOR). Para realizar a verificação de erros o transmissor calcula o CRC dos dados que vai transmitir e envia junto com os dados. O receptor recebe os dados, calcula o CRC e compara com o CRC enviado pelo transmissor. Se coincidirem, a probabilidade de acerto é 99,9969% [MEN2002].

O CRC USB oferece as seguintes probabilidades de detectar erros:

- Erro em único bit – 100%
- Dois bits errados – 100%
- Número par de bits errados – 100%
- Blocos de bits errados com tamanho inferior a 16 bits – 100%
- Blocos de bits errados com tamanho inferior a 17 bits – 99.9969%
- Outras condições de erros – 99,9969%

5.2.2 O PROCESSO DE ENUMERAÇÃO

"Processo de Enumeração" é a design ação dada ao mecanismo de configuração das funções e *hubs* conectados ao *hub* raiz, sendo composto por transações do tipo *control*. Ele é invocado na inicialização do computador ou no ato de inserção "on the fly "de funções. Com isso o Controlador Host pode:

- Atribuir identificadores (1 a 127) às funções; estes identificadores farão as vezes de endereço do dispositivo, que é o campo ADDR de 7 bits presentes nos pacotes do tipo *token*.
- Receber as características do periférico USB e de seu fabricante.

- Saber quais as modalidades de transações serão utilizadas nos ciclos de barramento para cada função.

A especificação USB exige que toda função implemente um subdispositivo que responda eletricamente às transações de controle (control). Tal subdispositivo recebe a designação de "*endpoint 0*". Todos os outros subdispositivos da função com propósitos gerais de I/O são endpoints com identificadores não nulos, ou seja, 0 < endpoint <=15, onde ENDP é o campo de 4 bits presente nos pacotes *tocken* [MEN2002].

5.2.3 DESCRITOR DE DISPOSITIVO

Um descritor de dispositivo, que serve para inicialmente identificar alguns atributos do periférico e de seu fabricante tem basicamente a seguinte forma:

COMPRIMENTO	1 BYTE
TIPO	1 BYTE
VERSÃO USB	2 BYTES
CLASSE	1 BYTE
SUBCLASSE	1 BYTE
PROTOCOLO	1 BYTE
TAMANHO EP0	1 BYTE
ID FABRICANTE	2 BYTES
ID PRODUTO	2 BYTES
VERSÃO	2 BYTES
NOME DO FABRICANTE	1 BYTE
NOME DO PRODUTO	1 BYTE
NÚMERO SERIAL	1 BYTE
NÚMERO DE CONFIGURAÇÃO	1 BYTE

TABELA 13 – DESCRITOR DE DISPOSITIVO

Comprimento: Informa o número total de bytes do descritor, que é igual a 18;

Tipo: O tipo do descritor;

Versão: Indica a versão do dispositivo USB – 0101h = versão 1.1 e 0200h é 2.0;

Classe, subclasse e protocolo: definem o tipo de periférico (teclado, mouse, etc). Isso permite que o sistema operacional carregue um device driver de acordo com os códigos presentes nesses campos, não necessitando desenvolver um driver específico.

EP0: Define o limite para o número de bytes que pode ser enviado ou recebido em cada pacote de dados endereçado ao *endpoint 0*. O tamanho minimo e mais usado é 8.

ID Fabricante: Valor fornecido para as empresas interessadas em criar dispositivos USB. Precisam enviar um email para admin@usb.org para receber o seu ID.

Nome do Fabricante, Nome do Produto e Número Serial: São índices que devem ser usados pelo *device driver* para apontar para *strings* alfa-numéricas específicas.

Número de Configurações: Define o número de descritores de configuração de que o dispositivo necessitará [MEN2002].

5.2.4 DESCRITOR DE CONFIGURAÇÃO

O descritor de configuração não define por si só todas as informações a serem passadas ao Controlador Host. Porém, as vezes são necessários outros pequenos descritores. O tamanho em bytes de todos os descritores deve de ser armazenado num campo denominado "comprimento total".

Entre outras coisas, esse descritor informa , por exemplo, a carga máxima de energia elétrica que o dispositivo necessita, sendo esse valor limitado a 250 ou equivalente a 500 mA. Essa é a carga máxima que trafega pelo cabo USB.

5.2.5 DESCRITOR DE INTERFACE

É uma das possíveis estruturas de dados que podem ser transmitidas logo após o descritor de configuração. Ele tem a sua utilidade quando o dispositivo USB é um protótipo e não possui driver próprio.

5.2.6 DESCRITOR HID (HUMAN INTERFACE DESCRIPTOR)

A vantagem de se usar a classe HID é não ser preciso desenvolver um device driver específico para o protótipo, já que o sistema operacional possui um driver com funções básicas de I/O para a classe HID.

5.2.7 DESCRITOR DE ENDPOINT

O campo endereço de endpoint define qual dos 15 possíveis valores de endereço (campo ENDP transmitido no token IN e OUT) será definido pelo descritor (o endpoint 0 é reservado para o controle).

Comprimento	1 BYTE
Tipo	1 BYTE
Endereço do Endpoint	1 BYTE
Atributos	1 BYTE
Tamanho máximo do pacote	2 BYTES
Intervalo de pooling	1 BYTE

TABELA 14 – DESCRITOR DE ENDPOINT

5.3 PACOTE USB-ETHERNET NO LINUX

O sistema operacional LINUX possui suporte total a USB através de bibliotecas e incorporadas ao seu Kernel. O pacote USB-ethernet é o suporte necessário quando precisamos conectar estações em rede.

5.3.1 INICIALIZANDO OS PACOTES USB-ETHERNET

O pacote USB-ethernet não é exclusivo para um hardware específico. Requisita certas funcionalidades e necessita ter um hardware USB suportando o device driver. Deve ter além disso dois endpoints para transferências Bulk entre o host e o periférico, um para cada direção; deve ter também um controle de endpoint, embora claro que isso esta implícito no hardware USB.

Entretanto, o hardware USB-Slave permite fornecer mais endpoint do que o mínimo necessário para o suporte ethernet. Alguns desses endpoints forçam o seu uso por outros pacotes, enquanto outros endpoints podem ser usados diretamente por outras aplicações, ou não são necessários existirem para construção de periféricos. Há outras possibilidades para que um periférico USB suporte múltiplas configurações, sendo que

o suporte ativo Ethernet é somente uma dessas configurações. O pacote USB-ethernet não tem autonomia sobre muito disso, porem confia no código de alto nível para informar se cada endpoint pode ser usado, além de outras informações. A função responsável por isso é a usbs_eth_init.

O primeiro argumento identifica e especifica a estrutura de dados usbs_eth afetado. Isso é esperado já que a grande maioria das aplicações afetadas querem somente providenciar um simples dispositivo USB-ethernet para um host simples, e os pacotes providenciam automaticamente uma adequada estrutura de dados usbs_eth0 para esse suporte. Se múltiplas estruturas usbs_eth são necessárias por alguma razão, então se faz necessário ser instanciado por outro código, e cada um necessita ser inicializado por uma chamada por usbs_eth_init().

Os próximos três argumentos identificam os endpoints que podem ser usados para comunicação USB: endpoint de controle, endpoint de recebimento para pacotes partindo do host para o periférico e o endpoint de transmissão para pacotes ethernet trafegando em outras direções. Obviamente os três endpoints devem de ser fornecidos por algum hardware USB. O pacote USB-ethernet assume que ele tem o acesso básico para receber e transmitir endpoints, objeto de uso dos controles de função usbs_eth_disable e usbs_eth_enable.

O pacote também assume que outro código não é interessante na mudança de estado USB ou mensagens de controle de classes: devemos instalar handlers para a mudança de estado e também para a mudança de controle nos endpoints. No final, também temos um argumento responsável pela indicação e obtenção do endereço MAC (ou endereço da estação ethernet). Não é possível obter o MAC com o pacote USB-ethernet. Porém, usamos um código de alto nível pode resolver isso e também vale uma verificação do cenário em que o pacote USB-ethernet esta sendo utilizado.

O dispositivo USB deve ser iniciado primeiro quando uma conexão entre o host e o periférico será estabilizada imediatamente e o driver do dispositivo contata no host o pacote USB-ethernet para identificar o endereço MAC.

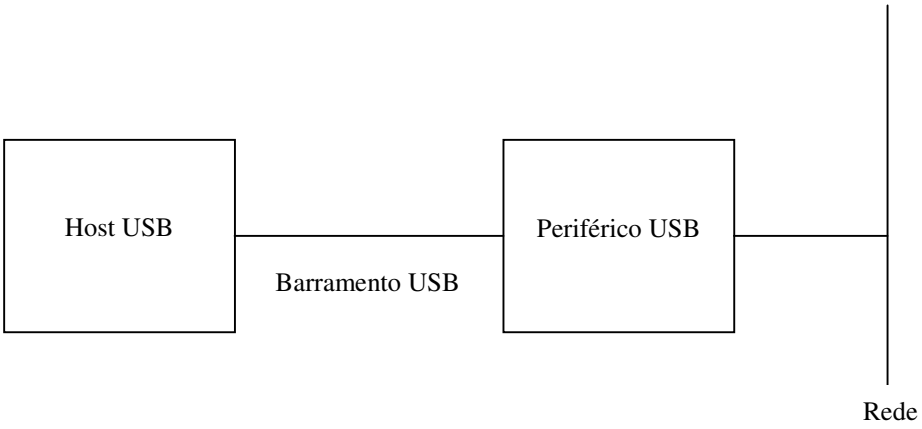

Figura 12 – Ligação USB em redes sem protocolo TCP/Ip

5.4 ENVIANDO PACOTES ETHERNET COM O HOST USB

Os pacotes USB-ethernet suportam dois modos de operação: No primeiro modo, suportado pelo Driver usbeth.h para usar com uma pilha TCP/IP rodando sobre um periférico USB. Todos os pacotes ethernet que chegam devem ser passados para a pilha TCP/IP, e somente a pilha determina a saída dos pacotes. Separando dessa inicialização é possível o controle da operação, e códigos de alto nível podem decidir não interagir com o pacote USB-ethernet diretamente.

No segundo modo não há uma pilha TCP/IP rodando sobre o periférico USB. Por exemplo, uma simples conversão USB-ethernet com um chip ethernet e uma porta USB: pacotes ethernet recebidos por um chip ethernet necessitam ser despachados para o host USB, e pacotes ethernet enviados pelo USB host necessitam ser enviados para fora do chip ethernet. **usbs_eth_start_rx** e **usbs_eth_start_tx** permitem o acesso em baixo nível para os pacotes USB ethernet .

Os dois modos de operação são mutuamente exclusivos . Se o modo de operaçao enthernet é ativado, então o código da aplicação deve ser comunicado como nível TCP/IP, e não pode ser usada a função de baixo nível. Em vez disso, o device driver de rede deve usar essas funções, assumindo que este tenha acesso exclusivo. O pacote não executa nenhum bloqueio.

O recebimento e transmissão das funções de trabalho em grande quantidade usam o mesmo modo. O primeiro argumento identifica a estrutura do USB-eth que esta sendo usada. Para maioria das aplicações, esta deve ser o usbs_eth0. O segundo

argumento especifica a locação do pacote ethernet, enviando para o **usbs_eth_start_tx** e chega para o **usbs_eth_start_rx**. Esse buffer deve corresponder com o protocolo USB-eth.

Pacotes de saída podem ter até 1516 bytes, consistindo em dois bytes identificando o cabeçalho para a criação USB-ethernet por um frame ethernet padrão (um cabeçalho com 6 bytes de endereço destino, 6 bytes de endereço de origem e dois bytes adicionais, restando então 1500 bytes para transporte). Os dois bytes do cabeçalho USB-ethernet consistem simplesmente no tamanho do frame ethernet, o tamanho restante do pacote não é incluído nesse cabeçalho, com o primeiro byte mais significativo

Para pacotes recebidos e substituídos devem ter ao menos 1516 bytes. Em casos ou circunstancias especiais na qual um pequeno buffer força ser salvo, por exemplo, quando o device driver host é modificado para suportar somente pequenos pacotes. Uma vez que o pacote é recebido o buffer determina dois bytes específicos para o USB-ethernet, sendo aceito como um frame normal. O cabeçalho empacota o tamanho do frame ethernet, excluindo o cabeçalho pelo primeiro byte mais significativo.

Ambos, **usbs_eth_start_tx** e **usbs_eth_start_rx** são assíncronos. A transferência é iniciada, e passado algum tempo, uma função de complementação deve ser invocada. O terceiro e quarto argumentos para o ambos, **usbs_eth_start_tx** e **usbs_eth_start_rx**, suprem a conclusão e argumentação da função respectivamente. A função completa pode ser invocada com 3 argumentos: um ponteiro para a estrutura de dados usbs_eth, normalmente usbs_eth0; e o suprimento completo dos dados; e o retorno do campo de código.

Um valor negativo indica que ocorreu um erro, por exemplo, EPIPE se a conexão entre o USB host e o periférico for rompida. Ou então EAGAIN se um endpoint estiver ‘travado”. Um valor positivo indica o tamanho total da transferência, quais devem corresponder com o tamanho no cabeçalho USB-ethernet mais uma adição de dois bytes para o cabeçalho próprio.

Se a transferência de dados for bem sucedida então a função de complemento deve ser invocada no contexto DSR preferencialmente no contexto da thread embora isso dependa da implementação do device driver USB utilizado. Entretanto a função completa é restrita em quais podem usar, em particular, e não devem fazer muitas chamadas que quiserem ou desse modo possibilitem bloqueios, alteração ou locação de memória. Observamos que o encerramento da transferência é rápida quando a função completa é invocada antes do retorno de **usbs_eth_start_rx** ou **usbs_eth_start_tx**.

Isto é particularmente desejável para auxiliar as threads que são desordenadas após iniciar a transferência de dados mas antes retornam para essas funções. Para operações de transmissão, é possível para **usbs_eth_start_tx** invocar toda a função imediatamente.

Se não houver uma conexão entre o host e o destino então a transmissão falhará imediatamente com –EPIPE. Em adição ao pacote USB-Ethernet é necessário checar o endereço destino MAC e assegurar que o frame ethernet é realmente o do host: seja qual for o endereço especificado na inicialização da chamada, ou pode ser um pacote em broadcast, ou o host deve estar habilitado no modo promiscuo.

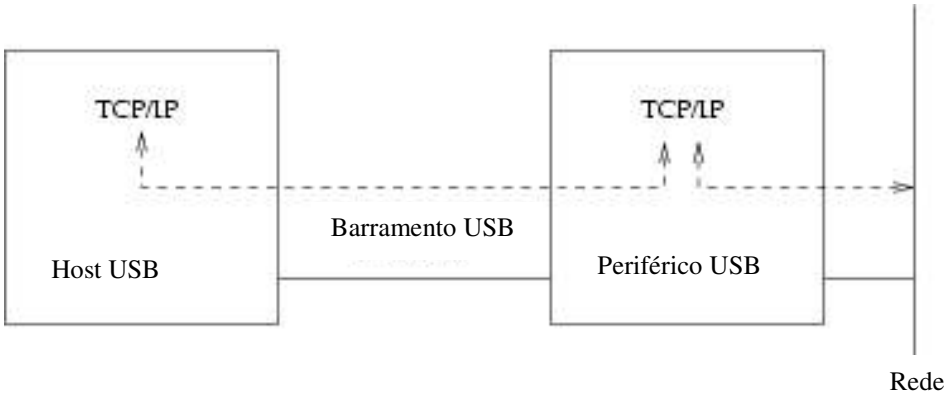

FIGURA 13 – LIGAÇÃO USB EM REDES COM PROTOCOLO TCP/IP

5.5 HARDWARE USB PARA MONTAGEM DE MULTICOMPUTADOR

Sabemos que para montar um Cluster de computadores são necessários meios físicos de comunicação entre os Nós que compões o mesmo. Até o final do ano 2000 haviam poucos periféricos que permitiam conectar dois computadores pela interface USB dos computadpres.

FIGURA 14 – SWITCH USB 200 MBPS

Haviam motivos de sobra para esse baixo interesse, pois o padrão USB 1.0 e 1.1 apresentavam taxas de Bandwitch com um máximo 12 Mbps e também o pouca interesse da indústria em adotar o padrão nas arquiteturas de Mother-Boards. Porém, com o lançamento da versão 2.0 o cenário se inverteu e os fabricantes de hardware e integradores começaram a lançar vários periféricos e dispositivos para USB, dando um grande impulso a tecnologia.

Para o ambiente de redes, também estão sendo lançados vários dispositivos de conexão, incluindo Hub's, Switch, Cabos e adaptadores USB-Ethernet, facilitando em muito o trabalho de montagem de redes de pequeno porte.

FIGURA 15 – ADAPTADOR USB-ETHERNET 10 MBPS

FIGURA 16 – CABOS E ADAPTADOR DUPLO USB

FIGURA 17 – CABOS USB COM PONTE DE LIGAÇÃO – PC TO PC

FIGURA 18 – HUB USB

6 BENCHMARCK DOS MEIOS DE CONEXÃO

O uso de CLUSTER de PC é uma solução de baixo custo e de fácil implementação, o que torna esse conceito bastante interessante para pequenas empresas ou organizações que necessitam de maior poder de processamento.

As técnicas para melhorar as ligações entre os nós de um CLUSTER podem variar de uma simples troca de cabos (por exemplo, Coaxial por Par-trançado), melhoria na qualidade das adaptadoras de rede utilizadas, chegando até a complexa implementação de um novo protocolo de comunicação, específica para esse ambientes operacionais. Porém, todas elas tem um limite de desempenho que se esgotam rapidamente, por um ou mais fatores, o que significa uma nova atualização e um maior custo operacional.

Entretanto, observando as novas tecnologias que estão emergindo, principalmente no que se refere a comunicação de dispositivos em PC, o padrão USB é uma tentadora oferta de ganhos de performance, o que se traduz numa opção bastante razoável para ligar nós de um CLUSTER. Praticamente todas as placas-mãe atuais contam com no mínimo duas raízes de conexão USB, baseadas na versão 1.1 ou 2.0, que garantem taxas de transferências acima de 400 Mbps (mais ou menos 50MB/s) e freqüência acima de 48 MHz, fugindo dos gargalos impostos pelo barramento PCI.

Nesse projeto, foram exploradas as possibilidades de conexão em rede utilizando-se de ligações USB em máquinas heterogêneas, utilizando a versão 1.1 (12 Mbits/s), sobre o protocolo TCP/Ip e Linux.

6.1 PLATAFORMA UTILIZADA

A implementação de CLUSTER pode ser feita com qualquer plataforma de software e hardware disponíveis no mercado. Para o presente projeto foram utilizados as seguintes plataformas:

2 Computadores Pentium III 550 MHz – Coopermine 64 MB RAM

Disco rígido Fujitsu 8.4 Gigabytes UDMA - 33 Mhz

Adaptadora de rede NE2000 10 Mbps

Sistema operacional Linux RedHat 7.3 – Kernel 2.4.18-3

Cabo USB A-A Bridge Pc to PC 5 mts

Cabos UTP-5 Par-trançado – 5 mts

Adaptador USB-Ether Dlink 1.1 10Mbps

Switch Surecom 8 portas EP-808X-R 10/100

6.2 METODOLOGIA DE TESTES

A bancada de testes foi montada inicialmente para coletar números de desempenho utilizando as adaptadoras Ethernet . Após essa avaliação preliminar, seguiu-se os testes com os cabos USB . Ambos foram realizados com o mesmo software, aplicando a mesma carga de trabalho aos 2 ambientes testados. Foram feitas 10 repetições em cada ambiente, retirando-se a média das mesmas em KB/s.

O número de repetições serve para garantir o grau de confiança das amostras de valores recolhidos nos testes.

6.3 RESULTADOS OBTIDOS

Os resultados obtidos seguiram a metodologia de testes descrita. As repetições foram feitas entre os computadores utilizando-se pacotes de 64 bytes transferidos por uma aplicação implementada em código C no ambiente Linux (NST Ethernet Benchmark).

Conexão	**KB/s Máximo**	**KB/s Média**	**Latência - Média**
USB-USB 1.1 12 Mbps	1536	1230	15 Milisegundos
ETHERNET 10 Mbps	1220	1136	4 Milisegundos
USB-ETHERNET 12 Mbps	1536	1080	18 Milisegundos

TABELA 14 – COMPARATIVO DE DESEMPENHO MÁXIMO E MÉDIO DOS ADAPTADORES

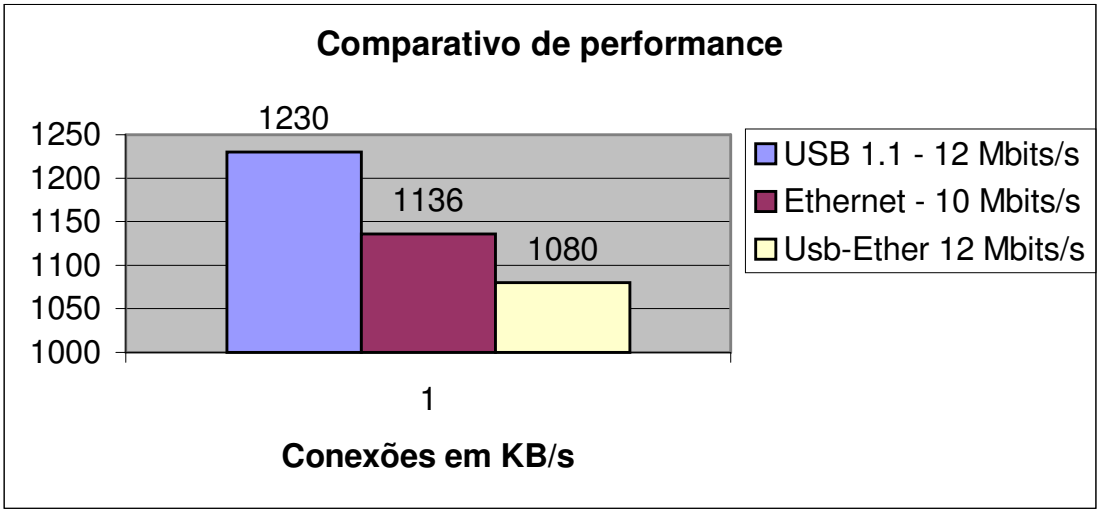

Gráfico 01 – Comparativo de desempenho entre os 3 tipos de adaptadores

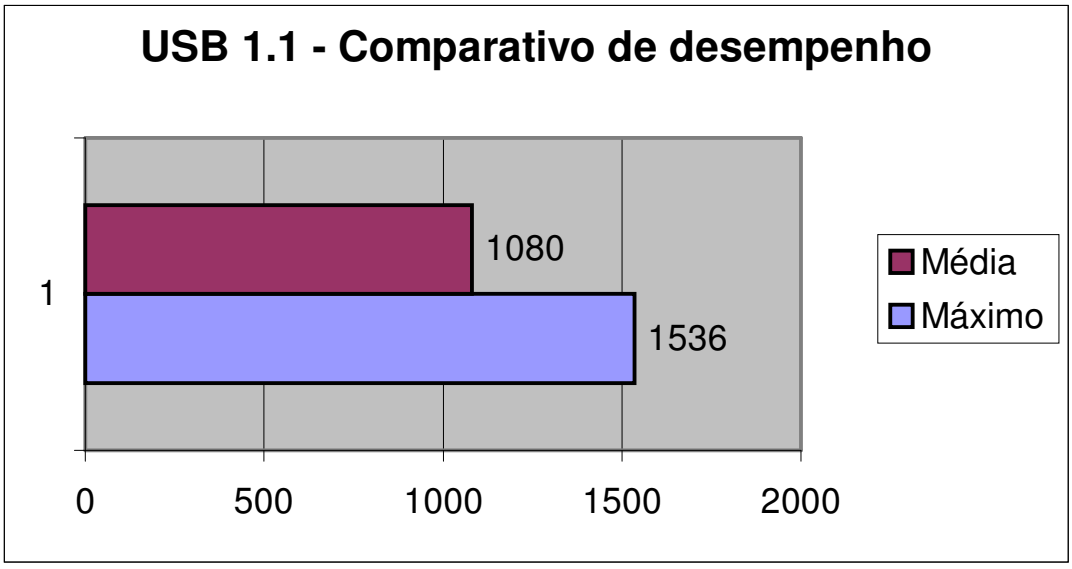

Gráfico 02 – Taxas de transferência Adaptador USB / Ethernet

6.4 PROBLEMAS DE LATÊNCIA

Verificando os números obtidos após os testes, confirma-se a característica do barramento USB de possuir tempos de latência bastante altos. A explicação para esses índices podem ser explicadas pela sua arquitetura de projeto.

A latência é uma das métricas fundamentais para medir o desempenho de tecnologias de interligação. A latência da comunicação é dada pelo tempo total necessário para transmitir uma mensagem de uma maquina origem a uma destino, e consiste de quatro componentes: (a) o overhead do software associado ao envio e a

recepção das mensagens, (b) ao atraso do canal, dado pela razão entre o tamanho total da mensagem e a largura de banda do canal, (c) o atraso de roteamento, causado pelo tempo gasto para encaminhar os pacotes a cada switch ao longo da rota e (d) o atraso de contenção, causado pelo tráfego que compete pela banda da rede. A partir do conceito de latência de comunicação é possível definir latência de rede, que corresponde à soma do atraso do canal e do atraso de roteamento.

Nesse ponto o USB sofre com uma latência mais alta, pois o mestre do barramento, inicialmente, é o próprio PC, transferindo essa responsabilidade para o primeiro dispositivo (HUB e Switch, no caso) ligados ao Host USB. Quando existe um periférico ligado ao Hosts USB, o barramento faz uma verificação a cada 2,5 ì s para se certificar que o Hub continua conectado, e esse por sua vez realiza a mesma verificação para conferir se os periféricos ligados a ele também continuam presentes. Fazendo um cascateamento de hub ou switch, essa verificação deve ser realizada independente de haver ou não comunicação.

Avaliando esse comportamento, fica fácil visualizar o problema de roteamento que o barramento impõem ao conjunto de periféricos ligados a ele. Esse comportamento também interfere na banda da rede, pois a constante verificação do canal de comunicação para saber se o periférico continua conectado ao barramento implica na transferência de mensagens, consumindo em média **10 %** da banda disponível (valor médio apurado nos testes) e isso representou 1,2 Mbps de toda a banda disponível de um total de 12 Mbps.

7 CENÁRIOS DE IMPLEMENTAÇÃO NO AMBIENTE CRUX

Após a realização dos testes de desempenho das conexões, podemos montar alguns cenários possíveis de implementação do CRUX. Esses cenários representam opções viáveis e possíveis de serem implementadas, tento como base a arquitetura do modelo atualmente implementado no Laboratório de Computação Paralela e Distribuída (LacPad) da UFSC.

7.1 CENÁRIO A -MODELO CRUX USB – USB

Esse cenário descreve o CRUX totalmente conectado por USB. O NC (Nó de Controle) possui um HOST-USB e duas saídas de conexão (sem divisão de canal). Uma delas esta ligada ao Crossbar USB e a outra ao barramento de serviço USB utilizando-se cabos USB modelo A-B com tamanho máximo de 5 Mts.

Essa ligação também pode ser feita utilizando-se uma única saída USB e um duplicador (Figura 10) mas com a desvantagem de haver uma divisão do canal de comunicação e respectivamente a sua banda passante. Cada NT (Nó de Trabalho) também possui um HOST-USB com duas saídas, sendo eles ligados ao Switch-USB por um cabo USB A-B e ao HUB USB também por um cabo USB A-B de 5 Mts de comprimento máximo cada.

No cenário proposto teremos 4 NC e 1 NT conforme descrito na Figura 5. O NT estará conectado ao Switch USB (Crossbar) por uma ligação USB (USB1), que será referenciado como endpoint0 e será o mestre do barramento nessa ligação, que vamos denominar de LNC1. O NT também estará conectado ao HUB USB (Barramento) através da segunda ligação USB (USB2), que também será referenciado como endpoint0 e será o mestre do barramento nessa ligação, que chamaremos de LNC2..

Na LNC1 teremos uma identificação para cada NT ligado ao Switch USB (Crossbar) que obedecerá a seguinte ordem: endpoint1 (NT1), endpoint2 (NT2), endpoint3 (NT3) e endpoint4(NT4). Durante as comunicações entre o NC e os NT's, o barramento USB usará essa identificação para referenciar os mesmos.

Já na LNC2 teremos uma configuração similar. O HUB USB (barramento) será o endpoint0 e cada um dos NT's será identificado na seguinte ordem: endpoint1 (NT1),

enpoint2 (NT2), endpoint3 (NT3), endpoint4 (NT4). Essa identificação é que será reconhecida pelo NT através do barramento USB.

A limitação que teremos nessa configuração é a quantidade de estações que podem ser conectadas ao Crossbar e ao barramento, que no caso é de 15 dispositivos (ou no caso NT's). Podemos fazer um cascateamento de HUB's, o que expandiria a capacidade de inserção de novos NT's, mas limitados a um total de 127 dispositivos (HUB) com 16 endpoints em cada um deles.

Esse é um cenário possível de ser implementação porém teria uma limitação de 12 Mbits em todos os canais físicos de comunicação utilizando-se o padrão USB 1.1, ou então na versão USB 2.0 no NC e NT, não seria impraticável alcançar taxas de transferência máximas próximas de 200 Mbits (limitadas pelo Crossbar USB). Em relação aos custos de implementação do modelo, ele seria mais barato que o modelo CRUX original, pois o mesmo demanda 2 adaptadoras de rede em cada NT e no NC, além do Hub e Switch Ethernet.

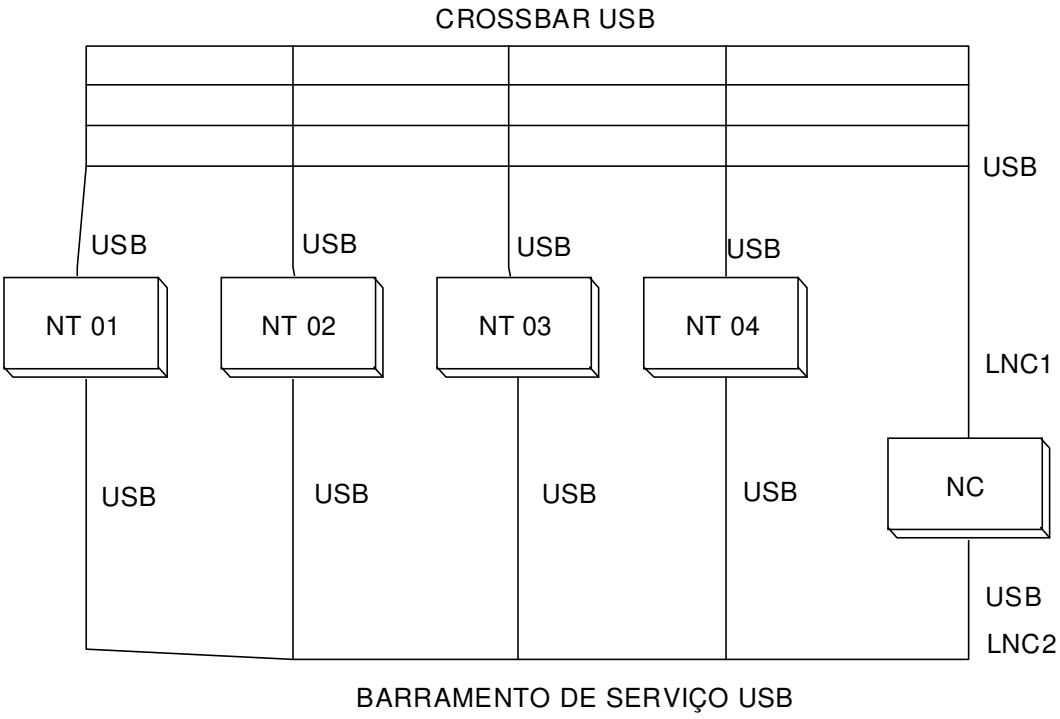

FIGURA 18 – CENÁRIO CRUX USB

7.2 CENÁRIO B - MODELO USB -ETHERNET

O segundo cenário descreve o CRUX conectado por adaptadores USB-Ethernet. O NC (Nó de Controle) possui um HOST-USB e duas saídas de conexão. Uma delas esta ligada ao Crossbar Ethernet utilizando-se adaptadores USB-Ethernet aos cabos Par-trançado. A ligação do NC ao barramento de serviço também terá adaptadores USB-Ethernet e cabo par-trançado UTP.

Cada NT (Nó de Trabalho) também possui um HOST-USB com duas saídas, sendo eles ligados ao Crossbar por um adaptador USB-Ethernet, e ao barramento de serviço através de um adaptador USB-Ethernet e um cabo UTP do adaptador até o barramento de serviço.

A implementação física desse cenário seria mais simples de implementar, pois a única referência que o NC terá de fazer é transferir para o adaptador USB-Ether o controle do barramento (mestre do barramento) e esse assumiria a função que antes pertenceria a adaptadora de rede. Esse controle se repetiria tanto na LNC1 quanto na LNC2.

Já os NT's fariam o mesmo tratamento para os adaptadores USB-Ether ligados aos suas interfaces USB. A diferença mais significativa desse cenário é que ao invés do Crossbar (Switch USB) e o barramento de serviço (HUB USB) nomear os NT's como enpoints, a referência seria ao adaptador USB-Ether que seria identificado como enpoint0 em todas as ligações.

O funcionamento do crossbar e barramento de serviços ethernet não seriam afetados, mas teoricamente haverá uma penalização maior em tempos de latência. A explicação para esse aumento diz respeito ao trabalho que o adaptador USB-Ether terá em rotear as mensagens passadas em Upstream decodificando o endereço PID para um endereço de porta ethernet e também em Downstream, de uma porta ethernet para um endereço PID.

Ao contrário do modelo anterior, podemos concluir que essa configuração teria um custo mais elevado na implementação, pois seriam necessários adaptadores USB-Ether, cabos USB e cabos par-trançado, além do Switch e Hub Ethernet.

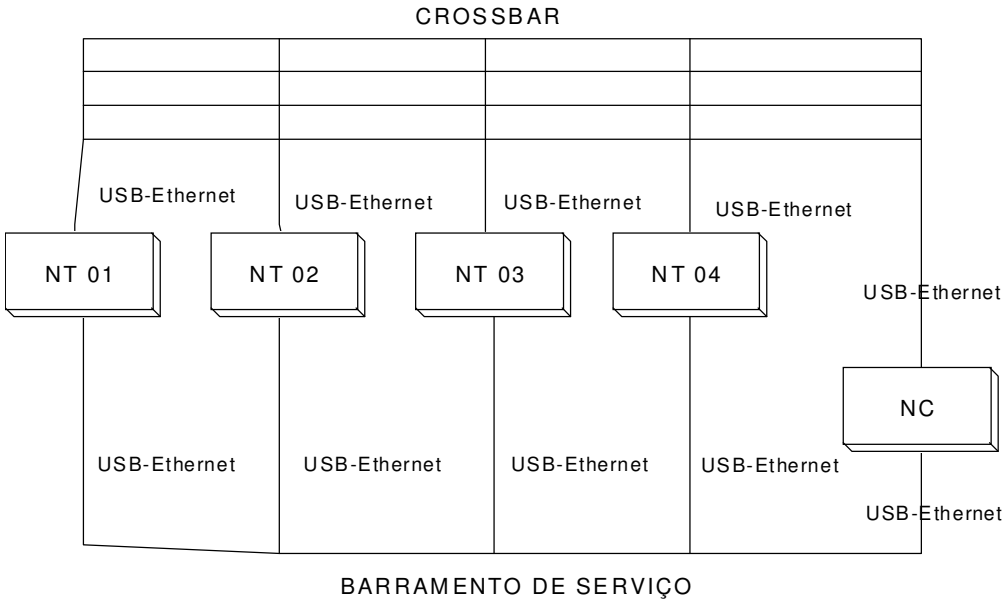

FIGURA 19 – CENÁRIO CRUX USB-ETHERNET

7.3 ANÁLISE GERAL DOS CENÁRIOS A E B

Comparando ambos os modelos propostos, o grau de eficiência dos dois pode ser observado através de algumas análises. Vejamos a seguir alguns argumentos que podem auxiliar na compreensão dos cenários, os dispositivos necessários e o impacto que eles podem gerar nas comunicações entre os NÓS.

Cenário A

1 hub USB com 4 portas

1 switch USB com 4 portas

10 cabos USB A-A

É o cenário mais interessante por utilizar todos os recursos padrão USB. Isso simplifica a forma de montar o CLUSTER e proporcionando também um menor custo. As aquisições seriam o HUB e o Switch USB, juntamente com os cabos A-A . Fazendo uma pesquisa de mercado, eses dispositivos teriam um custo inferior ao cabeamento, placas de rede, Switch e Hub ethernet. Utilizando todo o conjunto USB, os números relativos a taxa de transmissão de dados e tempo de latência terão um resultado bastante significativo.

Cenário B

1 hub Ethernet

1 switch Ethernet

10 cabos USB

10 adaptadores USB-Ether

Os elementos necessários para implementar o cenário B demonstram um custo mais alto para implementação. A vantagem nesse caso é aproveitar dispositivos ethernet (Switch e Hub) que por acaso já são utilizados pelo ambiente.

Outra dificuldade que vamos encontrar nesse cenário é a disparidade de performance entre os adaptadores USB-Ether e os dispositivos ethernet (Hub e Switch), uma vez que o padrão USB pode ter uma largura de banda variando entre 12 Mbps (USB 1.1) ou 480 Mbps (USB 2.0), comparado com os 100 Mbps do padrão ethernet. Os valores de ambos não são iguais, forçando um dos dois tipos a baixar a sua taxa de transmissão.

Podemos fazer uma análise e indicar o Cenário B com um valor de latência maior que o Cenário A. O Cenário A esta totalmente conectado via USB o que pode ser mais eficiente se for utilizado o padrão 2.0, o que garantiria uma taxa de transferência máxima teórica de até 200 Mbps. Já o segundo é misto e limitado a uma largura de banda de 100 Mbps, se forem utilizados adaptadores e conexões USB 2.0.

8 CONCLUSÃO

Os projetos que envolvem computação paralela e distribuída ocupam lugar de destaque nas universidades e centros de pesquisa no mundo todo. As técnicas e soluções encontradas para solucionar os principais problemas de performance, e em especial nos meios de comunicação podem envolver as mais diversas técnicas de otimização.

Dentre as soluções desenvolvidas, a utilização de interfaces de conexões do tipo USB podem ser utilizadas e foram avaliadas como muito significativas. A sua performance indica que o potencial desse barramento pode ser explorado, tendo a garantia de um bom retorno no desempenho de CLUSTER.

Apesar de ser um barramento criado e otimizado para dar suporte aos mais diversos tipos de periféricos como Scanner, Impressora, WebCam eetc, esta comprovado que o o USB pode ser utilizado na montagem de redes locais de pequeno porte. A topologia de conexão, os periféricos de expansão do barramento (Hub, Switch, adaptadores e cabos) e os protocolos disponíveis sustentam essa afirmação.

O desempenho geral relativo a taxas de transferência de dados também demonstram a versatilidade e eficiência do barramento quanto a largura de banda, comprovados através dos valores obtidos nos testes de comunicação. A latência ainda é um desafio a ser superado, problema esse que desagrada aos profissionais da área, mas que pode ser administrado com o desenvolvimento de software de alto nível, através da API disponível pelo USB.

Além dos resultados obtidos nas conexões e comunicações, esta bastante evidente a tendência da industria de tecnologia de eliminar o barramento PCI das atuais arquiteturas de computadores e substitui-la por barramentos seriais mais eficientes, caso típico do USB, uma vez que ele dispensa esse barramento para prover serviços de comunicação aos seus periféricos.

O que reforça essa afirmação são os novos padrões que estão sendo desenvolvidos, entre eles o FireWire (similar aos USB), o Hipertransport (barramento serial no FSB das arquiteturas ATHLON) e mais recentemente o interesse pela

Infiniband, arquitetura de comunicação entre e intra-computadores que elimina os barramentos PCI e também Gigabit Ethernet).

As limitações atuais que os padrões Ethernet e Fast-Ethernet impõem nas estruturas de CLUSTER já não compensam pelo pouco desempenho que oferecem, levando-se em conta a demanda cada vez maior por índices de performance exigidas pelas novas tecnologias de comunicação, aplicações e também pela Internet.

Apesar do baixo custo de implementação, ficar limitado a uma largura de banda de 100 Mbps (Fast-ethernet) já não pode ser admintido como viável em muitos projetos de pesquisa que notoriamente exigem maior desempenho, obrigando os desenvolvedores a partir para soluções mais caras, como por exemplo Gigabit Ethernet.

Inicialmente, o que desabona o barramento USB é a Escalabilidade, que pode ficar comprometida a no máximo 127 periféricos. Porém, observando o tamanho dos CLUSTER atualmente implementados, observa-se que eles possuem um número de nós inferior a esse.

O padrão USB na sua versão 2.0 possibilita um leque bastante amplo para explorar todo o seu potencial, e com isso substituir de vez as adaptadoras de redes tradicionais nas ligações entre computadores, admitindo-se também que o padrão evolua em novas versões, com largura de banda mais elevada e maior eficiência no canal de comunicação entre periféricos.

BIBLIOGRAFIA

[AND2000] Anderson, Don **Usb System Architecture (PC System Architecture Series)** Inc. MindShare (Contributor), 2000

[BAR1998] BARRETO, M.E. NAVAUX, P.O.A. RIVIÉRE, M. Deck: **A new model for a distributed kernel integrating communication and multithreading for support of distributed object-orientede application with fault tolerance**. Neuquén, Argentina: IV CACiC – Congresso Argentino de Ciencias de la Computation, **Anale**s..., 1998.

[COR1996] CORSO, Tadeu B., **Ambiente para Programação Paralela em Multicomputador**. Florianópolis – SC: Relatório Técnico n.1, INE-UFSC, Novembro de 1993.

[FEN1981] FENG, Tse-Yun, **A Surver of Interconnection Networks**, IEEE Computer, v.12 n. 14, December 1981; p 12-27

[FLY1966}FLYNN, M. **Very High-speed Computing Systems**. In: IEEE. 1966, Proceedings...1966, v. 54, p 1901-1909.

[FOS1995] FOSTER, Ian T. **Designing and Building Parallel Programs Concepts and Tool for Parallel Software Engineering.** Reading: Addison-Wesley, 1995.

[MEN2002] MENDONÇA, Alexandre,Zelenoyski, **PC: Um Guia Prático de Hardware e Interfaceamento**, MZ Editora Ltda. Rio de Janeiro, 3 ª Edição

[MER1996] Merkle, C. **Ambiente para execução de programas paralelos escritos na Linguagem Superpascal em um Multicomputador com Rede de Interconexão Dinâmica**, Dissertação de Mestrado, CPGCC /UFSC 1996.

[PAL2002] PALM BRASIL, **Índice de termos técnicos para PalmTops**, disponível na Internet via WWW em 01/2002: http://www.palmbrasil.com.br/vocab/index.html

[PAT2000] PATTERSON, DAVID A.; Hennessy, John L., **Organização e Projeto de Computadores – A Interface Hardware/Software**, LTC, Seg. Edição, 2000.

[PFI1998] PFISTER, Gregory F. **In Search of Clusters.** Upper Saddle River: Pretince all PTR,1998. 2 ª Ed.

[TOR1999] TORRES, GABRIEL **Hardware de Computadores- Guia completo**, Makron Books, 1999.

[TAN1997] TANEMBAUM, ANDREW S., **Redes de Computadores**, Campus, 1997

[PC2002] **Revista Pc's, Artigo SMT**, Edição núm. 29, Lucano, Janeiro de 2002

2 LINUX

Se procurar-mos uma definição para o Sistema Operacional Linux com certeza vamos encontrar respostas do tipo “é um clone do UNIX...”. Essa definição de certa forma esta correta, pois em 1991 quando Linus Torvalds resolveu apresentar o Kernel Linux, afirmando que era só um projeto para aprender a programar num 386, ele já sabia do potencial por trás da sua criação. Apesar de no inicio contar com bastante código não-portátil em Assembly e C, a sua criação foi o ponto de partida para a explosão do Linuz como Sistema Operacional de domínio público, apoiado pela Internet, o canal que possibilitou a sua disseminação.

A sua estabilidade e compatibilidade é uma herança do UNIX, combinada com um número cada vez maior de colaboradores que tornaram possível ao Linux rodar em todas as plataformas PC disponíveis, com recursos interativos e de fácil manipulação.

Dentre várias características que o S.O possui, podemos destacar como principais:

- Multiusuário e Multitarefa
- Proteção de Memória
- Memória Virtual
- Capacidade de SMP
- Flexibilidade do POSIX
- Redes
- Estabilidade
- FreeWare