Protocolos de Alto Desempenho: Um Tutorial

Marcelo Dias Nunes **Otto Carlos Muniz Bandeira Duarte**
email: bill@coe.ufrj.br **email: otto@coe.ufrj.br**

Universidade Federal do Rio de Janeiro
COPPE/EE - Programa de Engenharia Elétrica
Caixa Postal 68504 - CEP 21945-970 - Rio de Janeiro - RJ - Brasil
FAX: +55 21 290.6626

Sumário

Antes da introdução das redes locais (LANs - *Local Area Networks*), metropolitanas (MANs - *Metropolitan Area Networks*) e de longa distância (WANs - *Wide Area Networks*), a banda-passante era um recurso limitado nos sistemas de comunicação, pouco confiáveis devido aos freqüentes erros de transmissão. Com o advento destas redes, a banda-passante aumentou consideravelmente, ao ponto de fazer com que a vazão disponível para aplicações seja agora limitada pelo pesado processamento dos protocolos de Transporte. Este artigo apresenta um tutorial de cinco protocolos de Transporte (ou com sua funcionalidade) representativos, projetados especificamente para ter seu processamento otimizado e simplificado: Datakit e URP, Delta-t, NETBLT, VMTP e XTP.

Abstract

Before the introduction of Local Area Networks (LANs), Metropolitan Area Networks (MANs) and Wide Area Networks (WANs), bandwidth was a scarce resource in communication systems, which were unreliable due to frequent transmission errors. When these networks appeared, bandwidth was considerably increased until the available throughput for applications became limited by the heavy-weight Transport protocol's processing. This paper presents a survey of five representative Transport (or Transport-functional) protocols, specifically designed in order to optimize and simplify their processing: Datakit and URP, Delta-t, NETBLT, VMTP and XTP.

1 Introdução

Nos anos 60, a banda-passante dos meios de comunicação era um recurso caro e escasso em redes de longa distância, baseadas principalmente em tecnologia telefônica. Especificamente, taxas de transmissão de centenas, no máximo milhares, de bites por segundo eram disponíveis. Além disso, a Camada Rede oferecia um serviço de relativa baixa qualidade à Camada Transporte, devido à freqüência dos erros de transmissão.

A principal atribuição de um protocolo de Transporte é garantir uma transferência de dados fim-a-fim perfeitamente confiável. A fim de atingir este objetivo,

o projeto de protocolos de Transporte levava em consideração a utilização mais eficiente possível da banda-passante disponível e implementava elaborados mecanismos de detecção e recuperação de erros, para compensar a baixa confiabilidade das redes sobre as quais deveriam funcionar.

Nos anos 70, foi introduzido o conceito de redes locais (LANs). Em particular, as redes Ethernet elevaram a banda-passante para 10 megabites por segundo (Mbit/s). Nos anos 80, surgiram as redes metropolitanas (MANs), que juntamente com as redes locais atingiram taxas de transmissão da ordem de 100 Mbit/s. Para os anos 90, espera-se que as redes de longa distância (WANs) alcancem velocidades de gigabites por segundo. Ou seja, a banda-passante deixou há muito tempo de ser o fator limitante. Agora, é o processamento do protocolo que impede uma aplicação de utilizar uma fração razoável da banda disponível. Mais ainda: a utilização crescente de cabos de fibra ótica como meio físico reduz consideravelmente as taxas de erro, que no novo ambiente são devidos principalmente à perda de pacotes causada por *overflow*.

Isso tudo leva os projetistas de protocolos de Transporte a se defrontarem com duas alternativas: modificar os protocolos existentes para minimizar o processamento ou projetar desde o início protocolos completamente novos, que utilizem eficientemente as altas bandas-passantes e as baixas taxas de erro características dos atuais sistemas de comunicação.

Neste artigo, serão apresentados alguns protocolos de Transporte (e outros não exatamente de Transporte, mas com sua funcionalidade) especialmente desenvolvidos para sistemas de comunicação de alto desempenho. Os diferentes mecanismos usados por estes protocolos para atingir seus objetivos são comparados numa tentativa de determinar quais deles seriam mais apropriados à nova realidade dos sistemas de comunicação.

Na Seção 2, são apresentados, em linhas gerais, o funcionamento e as técnicas implementadas pelos protocolos Datakit e URP, Delta-t, NETBLT, VMTP e XTP. Conclusões são apresentadas na Seção 3.

2 Alguns protocolos de alto desempenho

2.1 Datakit e o Receptor Universal

Datakit [1-3] é uma arquitetura de comunicações projetada nos laboratórios da Bell no final dos anos 70. O princípio fundamental na arquitetura do Datakit é a comunicação através de uma cadeia contínua de octetos (*stream*). Porém, a unidade de dados básica adotada foi um campo de nove bites, e não um octeto, pois o bite extra é necessário para identificação (dados ou controle).

A arquitetura Datakit é dividida em camadas que apresentam a funcionalidade das Camadas Física, Enlace e Transporte do Modelo OSI [4]. A Camada Enlace deve ser a parte mais eficiente da rede, sendo responsável pelo encapsulamento dos octetos em quadros, pela comutação destes quadros e pela multiplexação de diferentes conexões em uma mesma linha de transmissão. Quando erros de transmissão são detectados, a informação corrompida é simplesmente descartada. Desta forma, estes erros se apresentam como informação perdida a Camada Transporte, que é responsável pela sua retransmissão. A Camada Enlace utiliza os códigos 0 a 7 para seus octetos de controle;

a Camada Transporte, 8 a 63; a Camada Sessão, 64 a 127. No caso desta última, apesar da coincidência de nomes, não se trata da Camada Sessão definida no Modelo OSI, mas apenas uma camada que realiza a supervisão dos processos usuários.

Numa rede Datakit, os chamados "computadores de controle da rede" são dedicados exclusivamente ao estabelecimento e liberação de conexões, de forma transparente ao usuário, para o qual estes computadores de controle são estações como qualquer outra. Logo, há uma separação entre os protocolos da Camada Transporte e os protocolos usados para administração de conexões, simplificando bastante o projeto da Camada Transporte.

Infelizmente, uma arquitetura baseada em cadeias de octetos pode gerar um *overhead* significativo no processamento. A utilização de campos de nove bites como unidade de dados fundamental traz problemas de alinhamento em memórias convencionais, o que diminui a capacidade de processamento do receptor, na maioria das vezes o principal fator de restrição ao desempenho do sistema. O receptor perde mais tempo analisando os dados transmitidos do que o transmissor leva para enviá-los. Isto, aliado ao fato de que sistemas de comunicação de uma maneira geral apresentam seu desempenho drasticamente aumentado quando podem processar grandes blocos de dados, levou os projetistas do Datakit a desenvolverem o Protocolo Receptor Universal (URP - *Universal Receiver Protocol*). Protocolos que utilizam o Receptor Universal apresentam complexidade mínima no receptor, que funciona basicamente respondendo a comandos do transmissor. O que se pretende é que cada estação possa se comunicar com qualquer outra na rede. Para isso, o transmissor de cada estação deve funcionar corretamente quando conectado a um Receptor Universal remoto. Uma vez que todas as estações utilizam o mesmo algoritmo para o receptor, qualquer estação poderá transmitir dados para quaisquer outras. Logo, apenas a especificação do Receptor Universal seria padronizada, uma vez que diferentes estações que suportem diferentes serviços implementariam seus próprios transmissores.

O Receptor Universal manipula os dados octeto por octeto. Estes octetos podem ser dados propriamente ditos ou controle do protocolo. A Tabela 1 mostra uma lista dos octetos de controle enviados do transmissor ao receptor. O receptor responde enviando os octetos de controle mostrados na Tabela 2.

Tabela 1. Comandos do transmissor.

Nome	Descrição
SEQ(i)	Número de seqüência anexado aos dados e usado para controle de fluxo e de erros. $0 \leq i < 8$.
BOT	Marca o início da cauda do último bloco de uma mensagem.
BOTM	Marca o início da cauda de um bloco que não é o último da mensagem.
BOTS	Marca o início da cauda de um bloco de cadeia de octetos.
ENQ	Pedido de informação a respeito do estado atual do receptor.
INIT0	Inicializa o receptor no modo caracter.
INIT1	Inicializa o receptor no modo bloco.

Tabela 2. Comandos do receptor.

Nome	Descrição
ECHO(*i*)	Número enviado como reconhecimento positivo indicando que os dados foram transferidos ao usuário.
REJ(*i*)	Número enviado como reconhecimento negativo indicando o próximo número de seqüência a ser recebido corretamente.
ACK(*i*)	Número enviado em resposta ao comando ENQ para indicar qual o último número de seqüência recebido corretamente.
INITREQ	Pedido de inicialização.
AINIT	Reconhecimento da inicialização.

A Figura 1 mostra um esquema do Receptor Universal, que é dividido em três partes:

- Estágio 1: aplicado a cada octeto de informação recebido da rede. Depois de processados, os dados são colocados numa fila.
- Fila: uma fila fifo (*first in, first out*) é usada como *buffer* de dados e códigos de controle que ainda não foram entregues ao usuário. Esta fila tem duas partes internas: uma área de armazenamento, onde são mantidos os dados ainda não checados, e uma fila principal, que contém dados recebidos corretamente.
- Estágio 2: tira dados da fila principal e os entrega ao usuário.

Figura 1. A estrutura do Receptor Universal.

São definidos dois modos de operação para o receptor: modo bloco e modo caracter.

- Modo Bloco: neste modo de operação, espera-se que o transmissor envie blocos de dados seguidos de uma cauda que contém um delimitador de início, o número de octetos de dados e o número de seqüência do bloco. De acordo com o tipo de cauda empregado, um bloco pode ser de mensagem ou de cadeia de octetos. Ambos sofrem controle de erros e blocos incorretos são descartados. A diferença entre eles está na resposta do Receptor Universal à detecção de um erro. No caso de um bloco de mensagem perdido, o receptor rejeita todos os blocos seguintes até que uma retransmissão do bloco perdido seja corretamente recebida. No caso de um bloco de cadeia de octetos, a perda de um bloco não previne a aceitação dos blocos seguintes recebidos corretamente.
- Modo Caracter: neste modo de operação, espera-se que o transmissor envie rajadas de dados com um código de controle SEQ seguindo cada rajada. Neste modo, a área de armazenamento da fila não é usada, pois todos os dados são movidos imediatamente para a fila principal e daí para o usuário.

2.2 Delta-t

No final dos anos 70, foi desenvolvido nos laboratórios Lawrence Livermore o sistema operacional LINCS (*Livermore Integrated Network Computing System*), para integração de sistemas heterogêneos, incluindo computadores pessoais e supercomputadores. Havia a necessidade de um protocolo de Transporte de alto desempenho que oferecesse um suporte eficiente tanto para transações (pedido/resposta) em comunicações cliente/servidor, quanto para conexões destinadas à acesso remoto a terminais e transferência de grandes quantidades de dados. Surgiu então o Delta-t [5-7], cuja meta era permitir o envio de pedidos, respostas ou grandes *buffers* de dados com exatamente dois pacotes, um de dados e outro de reconhecimento. Nenhum outro pacote é necessário para abertura ou fechamento de conexões.

O protocolo Delta-t tem a funcionalidade de uma Camada Rede não orientada à conexão e de uma Camada Transporte, mas pode ser implementado sobre outros protocolos de Rede não orientados a conexão, como IP ou CLNP. A comunicação se dá entre entidades identificadas por endereços de 64 bites, únicos na rede, que podem ser endereços genéricos independentes do local ou endereços fisicos particulares a alguma estação.

O Delta-t foi um dos primeiros protocolos de Transporte a se preocupar com a segurança dos dados. Por isso, todos os pacotes contém dois campos, uma marca de proteção de quatro bites (não classificado, confidencial, secreto, etc.), de forma que mecanismos de segurança possam ser empregados em todos os nós da rede, e um número aleatório de 64 bites. Este número estabelece o direito do transmissor de enviar dados ou pacotes de reconhecimento. Também serve como identificador de transação para sincronizar pedidos e respostas.

No Delta-t, as unidades de dados visíveis na interface com o usuário consistem em bites e nos símbolos B, E e W. O bite foi escolhido pois na época em que o protocolo Delta-t foi projetado, máquinas de 36 e 60 bites eram comuns. Os protocolos das camadas superiores definem a maneira como os símbolos são usados. Geralmente, os símbolos B (*beginning*) e E (*ending*) são utilzados para delimitar o início e o fim de uma mensagem. O símbolo W pode ser utilizado pelo transmissor para indicar um ponto de ativação (*wakeup*), significando que já há informações suficientes para serem processadas pelo receptor.

Ao contrário do Datakit ou do URP, que não oferecem controle de fluxo, o protocolo Delta-t implementa um esquema de janelas deslizantes. Neste tipo de controle, uma janela de números de seqüência é mantida no transmissor. O receptor mantém um variável, em módulo, que representa o número de seqüência do pacote que pode ser aceito. Qualquer pacote que chegue fora da seqüência deve ser descartado. A janela sempre tem um tamanho fixo, e simplesmente desliza à medida que os pacotes são reconhecidos. Estes números de seqüência são usados para proteção contra perdas, duplicações ou dados fora de ordem. Também há um mecanismo opcional de *checksum* para detecção de dados corrompidos. Isto contrasta com a simplicidade do Receptor Universal, que detecta apenas dados perdidos.

Na arquitetura Delta-t, uma conexão representa o tempo em que informações de estado são mantidas em cada entidade comunicante. A abertura de uma conexão se refere à alocação confiável deste estado e o fechamento se refere à dealocação confiável

e não ambígua deste estado. O protocolo Delta-t realiza um gerenciamento de conexões livre de riscos, sem trocas de pacotes, ou seja, a chegada do primeiro pacote da comunicação abre uma conexão. Isto é chamado de estabelecimento implícito de conexão.

Para uma dada conexão, o transmissor e o receptor mantêm informações de estado e temporizadores, durante a transferência de dados. O temporizador do receptor assegura que o estado seja mantido por tempo suficiente para detectar todas as possíveis duplicatas, garantindo uma abertura de conexão confiável, e que a estação não aceitará pacotes num intervalo seguro em caso de queda de algum nó. O conjunto dos temporizadores de transmissão e recepção assegura que o estado seja mantido por tempo suficiente para garantir uma liberação clara da conexão e que números de seqüência aceitáveis sejam gerados. O temporizador do transmissor assegura que os números de seqüência não serão reutilizados, mesmo ocorrendo queda de algum nó, até que todos os pacotes de dados ou reconhecimento com um determinado número de seqüência tenham expirado. Como as conexões são mantidas por temporizadores, ou seja, depois de algum tempo sem nenhuma transferência de dados os temporizadores estouram e fecham automaticamente a conexão, a liberação de conexões também é implícita.

Finalmente, um campo de tempo de vida nos cabeçalhos dos pacotes é decrementado durante o roteamento, a retransmissão e o reconhecimento para assegurar que o período de existência de um pacote Delta-t seja limitado. Entretanto, isto aumenta o processamento dos nós da rede, que são obrigados a implementar mecanismos para ler cada pacote, decrementar seu tempo de vida e jogá-lo de volta para a rede.

2.3 NETBLT

O NETBLT (*Network Block Transfer*) [8,9] é um protocolo de Transporte especialmente projetado para atingir altas vazões em aplicações que envolvam a transmissão de grandes quantidades de dados. Este protocolo apresenta novas propostas de controle de fluxo e recuperação de erros em relação aos protocolos de Transporte convencionais e aos descritos nas seções anteriores.

Uma vez que a quantidade de dados a ser transferida pode ser muito grande, o usuário fornece um *buffer* de dados à entidade NETBLT transmissora, que o quebra em pacotes e os transmite à rede na forma de datagramas Internet. A entidade NETBLT receptora carrega estes pacotes num *buffer* do mesmo tamanho fornecido pelo seu usuário. Quando chega o último pacote do *buffer*, esta entidade verifica se todos os pacotes naquele *buffer* foram recebidos corretamente. No caso de perda de pacotes, a entidade NETBLT receptora solicita que eles sejam retransmitidos. Quando o *buffer* tiver sido completamente recebido, a Camada NETBLT notifica o usuário, que então libera o *buffer* anterior e fornece um novo para receber mais dados. A entidade NETBLT receptora notifica então ao transmissor que um novo *buffer* está pronto, e o transmissor prepara e envia o *buffer* seguinte da mesma maneira.

Uma novidade do NETBLT em relação aos protocolos anteriores é a utilização de duas estratégias para controle de fluxo: uma a nível de *buffer* e outra a nível de usuário. Antes que um *buffer* seja transmitido, a Camada NETBLT confirma que ambos os usuários estabeleceram *buffers* de mesmo tamanho, que um está pronto para enviar e o outro está pronto para receber dados. Qualquer usuário pode portanto controlar o fluxo de dados não fornecendo um novo *buffer*.

Uma vez que os *buffers* podem ser muito grandes, outro método de controle de fluxo deve ser utilizado durante a transmissão de um *buffer*. Como o NETBLT se propõe a oferecer alta vazão para suas aplicações, ele deve ser capaz de transmitir constantemente. Se houver espaço de armazenamento suficiente no módulo do protocolo, o controle da taxa de transmissão oferece essa funcionalidade, que é uma das principais inovações do NETBLT. Este controle opera independentemente do mecanismo de recuperação de erros, pois coloca os dados a serem retransmitidos na mesma fila com os novos dados. Logo, todos os dados deixam a fila à mesma taxa de transmissão. Controle de fluxo por taxa também pode reduzir problemas de congestão: já que as retransmissões de pacotes ocorrem misturadas aos novos dados, a rede não é sobrecarregada. Além disso, temporizadores de retransmissão são baseados na taxa de transmissão, não no tempo de ida e volta de um pacote. Portanto, eles podem ser mais precisamente estimados, resultando em menos retransmissões desnecessárias. Finalmente, a taxa pode ser ajustada dinamicamente para melhor refletir a capacidade da rede em um dado instante.

A taxa de transmissão é negociada pelas entidades NETBLT transmissora e receptora durante o estabelecimento da conexão e renegociada depois de cada transmissão de um *buffer*. O transmissor envia os pacotes em rajadas, num intervalo de tempo que também é negociado. O controle de taxa do NETBLT é dividido em duas partes, o tamanho da rajada e sua taxa de transmissão, sendo a razão tamanho/taxa igual ao tempo de transmissão médio por pacote. Intervalo da rajada é o tempo em milissegundos entre o início da transmissão de uma rajada e o início da próxima. Tamanho da rajada é o número de pacotes de dados numa rajada.

O envio de mensagens de reconhecimento (positivo ou negativo) é submetido à transmissão bem sucedida de um *buffer* ou à determinação, pelo receptor, que alguns pacotes de um dado *buffer* foram perdidos. No primeiro caso, uma única mensagem reconhece todos os pacotes em um *buffer*. No segundo caso, uma única mensagem avisa ao transmissor exatamente quais pacotes devem ser retransmitidos. É a chamada retransmissão seletiva. O protocolo NETBLT também torna a recuperação de erros mais eficiente colocando o temporizador de retransmissão no receptor. A qualquer momento, o receptor sabe exatamente que pacotes chegaram e quais não chegaram. Logo, quando ocorre o estouro do temporizador, o receptor pode eliminar retransmissões desnecessárias porque sabe exatamente de quais pacotes deve pedir retransmissão.

O controle de erros é feito através de três pacotes de controle: GO, RESEND e OK. Quando a entidade NETBLT receptora tem um *buffer i* pronto para receber dados, ela envia um pacote GO[*i*] para o transmissor. Este pacote é o sinal para o transmissor começar a enviar o *buffer i*. Tão logo o primeiro pacote de dados em *i* chegue no receptor, este dispara um temporizador associado ao *buffer i*. Este temporizador estima o tempo que levará para preencher o resto do *buffer*, baseado na taxa de transmissão previamente negociada. O receptor agora aguarda dois eventos possíveis: ou chegam todos os pacotes do *buffer* ou o temporizador estoura. No primeiro caso, o receptor interrompe o temporizador e envia um pacote OK[*i*], comunicando ao transmissor que ele pode liberar o *buffer i*. No segundo caso, o receptor verifica quais os pacotes do *buffer i* que chegaram e envia um pacote RESEND[*i*] contendo uma lista dos pacotes de dados perdidos. Assumindo que todos os pacotes retransmitidos chegaram corretamente, o receptor envia um pacote OK[*i*] e interrompe o temporizador.

Mesmo com um mecanismo de recuperação de erros eficiente, uma transmissão contínua em redes com longos atrasos não é possível sem a utilização de uma janela grande. Por esta razão, as implementações do NETBLT devem usar uma janela tão grande quanto os recursos disponíveis permitirem, de tal forma que dificilmente uma transmissão seja forçada a parar no limite da janela. Como se pode observar no mecanismo descrito no parágrafo anterior, a janela utilizada é de um *buffer*, não importando seu tamanho. Entretanto, janelas maiores, de vários *buffers*, podem ser implementadas.

2.4 VMTP

O VMTP (*Versatile Message Transfer Protocol*) [10-13] é um protocolo de Transporte projetado para uso por um grupo de computadores num sistema distribuído. Assume-se que a comunicação seja dominada por acesso a páginas de arquivos, chamadas remotas de procedimentos, datagramas em tempo real e comunicações ponto a multi-ponto.

O VMTP é basicamente orientado a transações (pedidos e respostas). Na nomenclatura VMTP, a entidade que solicita um serviço através de um pedido é chamada cliente, enquanto que a entidade que atende o pedido através de uma resposta é chamada servidora. Uma transação de mensagens é iniciada por um cliente enviando uma mensagem de pedido a uma entidade servidora e termina com o servidor enviando de volta uma mensagem de resposta. A resposta reconhece para o cliente o recebimento do pedido pelo servidor. O próximo pedido do cliente, um reconhecimento específico ou o estouro de um temporizador reconhece para o servidor o recebimento da resposta pelo cliente. Esta é a transação de mensagens básica no VMTP. O cliente só pode ter uma transação de mensgens não reconhecida de cada vez. Cada entidade é endereçada por um identificador de 64 bites. Um grupo de entidades pode ser identificado por um único identificador de entidade, mesmo que elas estejam distribuídas por várias estações.

Apesar do Delta-t não usar o termo “transação de mensagens”, fica evidente que esta operação nada mais é do que abertura e fechamento implícitos de conexão, em que o primeiro pacote de dados abre a conexão e um temporizador se encarrega de fechá-la automaticamente.

Na arquitetura VMTP, o servidor tem no máximo um descritor de conexão para cada cliente ativo. Este descritor é reutilizado se um novo pedido é recebido do cliente dentro de um certo intervalo de tempo e é automaticamente liberado mediante o estouro de um temporizador. De maneira semelhante, cada cliente exige apenas um descritor, independentemente do número de transações em que ele esteja envolvido, aproveitando-se do fato de que só pode haver uma transação de cada vez.

O protocolo VMTP oferece três variações da transação básica descrita anteriormente. Na primeira variante, a transação de mensagens de grupo, o cliente difunde um pedido para um grupo de entidades servidoras e pode receber várias respostas. É o suporte do VMTP a comunicações ponto a multi-ponto, uma das principais características que o diferencia dos demais protocolos descritos até aqui. A transação de mensagens é terminada pelo cliente iniciando outra transação. Na segunda variante, o cliente envia um pedido como um datagrama com uma indicação de que nenhuma resposta é esperada. O pedido pode ser enviado confiavelmente ou não (com ou sem reconhecimento). Na última variante, a mensagem de pedido pode ser

"expedida" para outro servidor, sendo que caberá àquele servidor responder diretamente ao cliente. Desta forma, o cliente não precisa saber que uma mensagem teve que ser rerroteada.

Até o VMTP, nenhum dos protocolos descritos neste artigo se preocupava com a implementação de esquemas eficientes de endereçamento. A estabilidade do endereçamento no VMTP é essencial para oferecer uma base adequada sobre a qual conexões a níveis superiores são estabelecidas. Por endereço estável, entende-se aquele que mantém o mesmo significado ou associação enquanto for válido. Um cliente pode usar o identificador de um servidor repetidamente e ter certeza que ou o identificador mapeará sempre no mesmo servidor ou então não é mais válido. Semelhantemente, o servidor pode comparar o identificador de um cliente com um anterior e ter certeza de que se trata do mesmo cliente se os identificadores forem iguais. Finalmente, o servidor pode testar a validade do identificador de um cliente, liberando os recursos reservados para aquele cliente, caso seja inválido. Os identificadores de entidades podem ser de dois tipos:

- identificadores estritamente estáveis: mantêm o mesmo significado ou associação indefinidamente uma vez que são atribuídos.
- identificadores T-estáveis: têm no máximo um significado ou associação válida num intervalo de tempo T, podendo ser alterados ao longo do tempo. Garante-se que um identificador de entidade T-estável não será reutilizado por pelo menos T segundos a partir do momento em que se torna inválido.

Há também os domínios de entidades, que são grupos fechados de entidades VMTP comunicantes. Entidades em domínios diferentes podem ter o mesmo identificador e apenas entidades no mesmo domínio podem se comunicar diretamente via VMTP. Porém, uma entidade pode participar de vários domínios simultaneamente. Domínios de entidades podem ser identificados por um número de 32 bites. Cada pacote contém este identificador, de forma que uma estação possa identificar os pacotes de seu domínio e ignorar todos os outros.

Uma implementação do VMTP deve tratar de duplicações de pacotes de pedido e resposta. Um pacote de pedido é identificado pelo identificador da entidade origem e por um identificador de transação de 32 bites. Este identificador é crescente com módulo 2^{32}, portanto rotativo. O receptor de uma nova transação de mensagens aloca um descritor de transação que armazena os identificadores da entidade origem e da transação, além de outras informações. Quando um pedido duplicado chega ao servidor antes que ele tenha gerado a resposta, a duplicata é facilmente detectada e descartada pelo módulo VMTP, através do descritor de transação. Quando um pedido duplicado chega depois da resposta ter sido gerada, mas antes do estouro do temporizador do descritor ou outra transação de mensagens do mesmo cliente, a resposta deve ser retransmitida. Normalmente, o servidor salva uma cópia da mensagem de resposta até que o temporizador do descritor estoure. Porém, para aumentar a eficiência, o servidor pode estabelecer que a transação seja idempotente. Neste caso, a retransmissão do pedido depois que a resposta tenha sido enviada chega ao servidor como um novo pedido. Isto elimina o *overhead* em se manter uma cópia da resposta para retransmissão.

Quanto aos pacotes de resposta, eles serão descartados pelo cliente se não "casarem" com o identificador de transação associado ao identificador da entidade cliente à qual são entregues, ou se o identificador do cliente for inválido. Uma resposta correta, entretanto, não será considerada duplicata se atender a algumas relações de

tempo. Especificamente, o intervalo de reciclagem, ou reutilização do identificador de transação deve ser maior que o tempo máximo levado pelo pacote de resposta para chegar ao cliente depois da última retransmissão do pedido. Além disso, a vida útil dos endereços T-estáveis deve ser grande o suficiente para garantir que não serão reutilizados por um intervalo de tempo seguro no caso de queda, ou *crash*, de uma estação.

Atualmente, o *hardware* das redes de comunicação tem evoluído para oferecer baixas taxas de erro e altas velocidades, de forma que as retransmissões são agora mais freqüentemente devidas a *overruns*. Quando pacotes são perdidos por *overrun*, erros sistemáticos são mais prováveis de ocorrer. Por exemplo, se um grupo de N pacotes é transmitido a uma velocidade ligeiramente superior a do receptor ou de algum nó intermediário, todo k-ésimo pacote poderá ser perdido. Logo, optou-se pela retransmissão seletiva, como no NETBLT.

Uma mensagem VMTP consiste de um cabeçalho de 76 octetos mais 16 koctetos de segmento de dados. Este segmento de dados é dividido em blocos de 512 octetos. Uma mensagem é encapsulada em um grupo de pacotes, ou seja, uma seqüência de pacotes com uma cópia do cabeçalho e zero ou mais blocos por pacote. O segmento de dados em cada pacote é representado por uma máscara de 32 bites. O receptor de um grupo de pacotes mantém uma máscara acumulativa dos blocos recebidos para uma dada transação. Se o temporizador estourar com um grupo de pacotes incompleto, um reconhecimento é enviado, indicando os blocos recebidos e disparando a retransmissão dos blocos perdidos. Portanto, o receptor pode solicitar a retransmissão de qualquer porção do grupo de pacotes.

O VMTP também utiliza controle de fluxo por taxa, à semelhança do NETBLT, mas o mecanismo adotado é diferente: “espaços”, ou intervalos, são introduzidos entre os pacotes de um grupo, para reduzir a taxa de transmissão. Clientes e servidores podem explicitamente comunicar os intervalos desejados, e podem fazer ajustes baseados nos pedidos de retransmissão seletiva. Além do mais, o transmissor pode periodicamente tentar reduzir o intervalo quando não ocorrem perdas de pacotes, para assegurar que está transmitindo à taxa máxima que o receptor pode suportar. Logo, o mecanismo de retransmissão seletiva provê a realimentação necessária para indicar que a taxa de transmissão está muito alta e também minimizar a perda de desempenho decorrente do *overrun*.

A Figura 2 mostra o formato de um pacote VMTP. Vários campos já foram mencionados ao longo da seção. Segue-se abaixo uma breve descrição dos restantes:

- Identificador de Autenticação - se a mensagem não estiver cifrada, indica a autenticação da entidade transmissora; caso contrário, é usado para identificar a chave correta para decifrá-la.
- Flags:
  1. EOM- fim de mensagem;
  2. SDA- segmento de dados presente;
  3. DGM- pedido de datagrama;
  4. CMD- entrega condicional de mensagem;
  5. IMA- reconhecimento imediato de mensagem;
  6. IPA- reconhecimento imediato de pacote;
  7. RES- reservado.

- Offset - indica o número do primeiro bloco do segmento de dados de um pacote, relativo ao início do segmento de dados da mensagem inteira.
- Máscara de Entrega - indica quais blocos da mensagem estão presentes no pacote.
- Função - indica o tipo do pacote (pedido, resposta, etc.).
- Bloco de Controle de Mensagem do Usuário - 32 octetos de dados do usuário, que podem ser utilizados, por exemplo, para passagem de parâmetros numa chamada remota de procedimento.

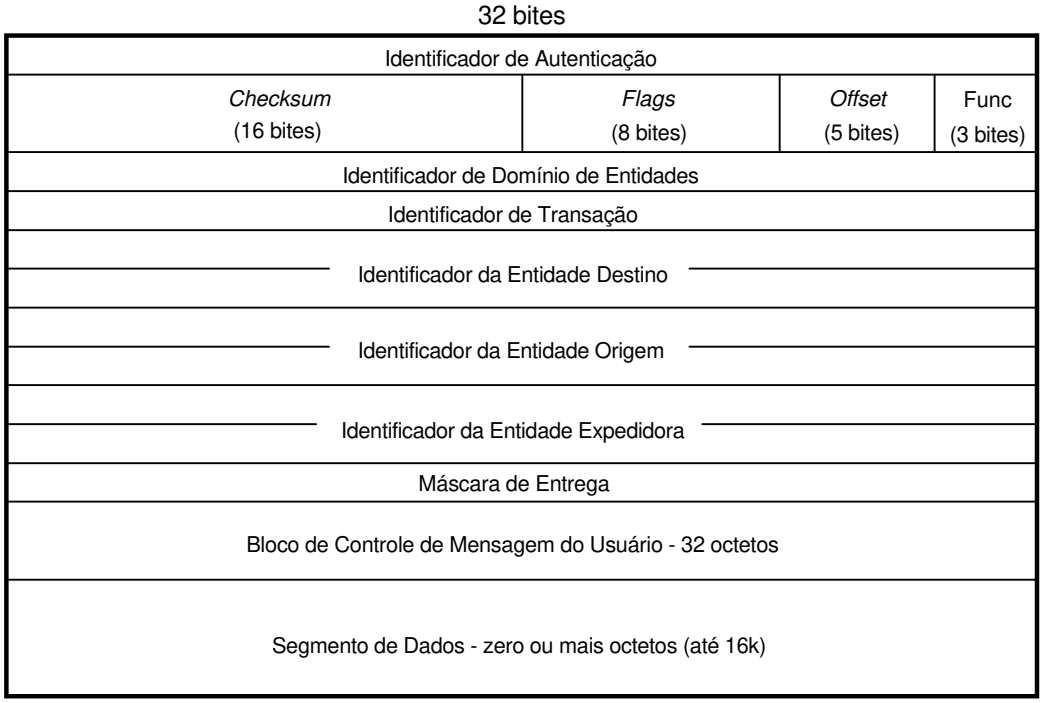

Figura 2. Pacote VMTP.

2.5 XTP

Em 1987, a equipe de projetos da Protocol Engines Inc., criada para desenvolver a implementação em silício de um protocolo de Transporte de alto desempenho, percebeu que modificar os protocolos existentes seria bem mais complexo que projetar um novo protocolo. Deste esforço, surgiu o XTP (*Xpress Transfer Protocol*) [14-16].

O XTP foi projetado para acomodar a realidade dos sistemas modernos: alta vazão e baixas taxas de erro. É um protocolo programável no sentido em que o transmissor pode selecionar as opções da comunicação, ou seja, o protocolo oferece mecanismos e o usuário seleciona políticas. Isto advém do reconhecimento de que apenas o usuário conhece suficientemente a aplicação para melhor otimizar os parâmetros da comunicação. O protocolo XTP incorpora características de todos os protocolos analisados anteriormente neste artigo, além dos protocolos TCP [17] e OSI TP4 [18]. Entretanto, não é exatamente um protocolo de Transporte, pois combina também funcionalidades da Camada Rede, como roteamento. Mas apesar de toda sua capacidade, ainda é baseado numa máquina de estados finita, visando implementação em silício.

O protocolo XTP apresenta diversas melhorias funcionais e de desempenho com relação aos protocolos tradicionais:

- Dados “fora da banda”: eventualmente, pode ser necessário enviar informações a respeito da conexão sem encaixá-los na conexão propriamente dita. No escopo deste artigo, o XTP é o único a oferecer este recurso, permitindo que cada pacote carregue oito octetos de dados “marcados”. Estes dados são entregues ao usuário final, mas não são interpretados pelo XTP.
- Estabelecimento e liberação de conexões: o XTP consegue confiabilidade com a troca de apenas dois pacotes. Este estabelecimento implícito de conexões pode porém ser substituído pela negociação habitual. O fechamento de uma conexão permite a implementação de várias políticas, como fechamento bidirecional, unidirecional e abortos. A não ser por esta última, as demais modalidades de desconexão são claramente negociadas. Não há nenhuma espécie de desconexão implícita por temporizadores.
- Cabeçalho e cauda: o cabeçalho de um pacote XTP contém informações de controle, enquanto que o *checksum* é colocado na cauda. Desta forma, o *checksum* poderia ser calculado por uma unidade de *hardware* independente e anexado aos dados à medida que fossem sendo transmitidos.
- Campos de comprimento fixo: por uma questão de eficiência, o protocolo XTP fixa o comprimento do cabeçalho e da cauda e a posição de todos os *flags* no cabeçalho.
- Reconhecimento: o protocolo XTP permite que o transmissor decida se e quando reconhecimentos são desejáveis, ou seja, o controle é do transmissor.
- Tamanho da janela: sua janela deslizante é de 32 bites, ou seja, permite até 4 bilhões de octetos não reconhecidos em cada conexão. Poderá passar facilmente para números de seqüência de 64 bites.
- Controle de taxa: permite controle sobre a taxa de transmissão dos dados e a quantidade de dados transmitidos. É o mesmo mecanismo do NETBLT. Porém, diferentemente daquele protocolo, o controle de taxa no XTP é utilizado para regular a capacidade de processamento de cada nó da conexão, não para controle de fluxo.
- Recuperação de erros opcional: algumas aplicações não exigem a retransmissão de dados perdidos. No XTP, o esquema de retransmissões pode ser desligado.
- Retransmissão: o protocolo XTP implementa o mecanismo *go-back-n*, em que a janela de transmissão é reinicializada a partir do pacote perdido, mas também implementa retransmissão seletiva como no NETBLT e no VMTP, em que o receptor fornece um vetor indicando as seqüências de informação corretamente recebidas, de tal forma que o transmissor possa retransmitir apenas as lacunas.
- Prioridades: um problema com protocolos de Transporte convencionais é que eles não são muito receptivos a dados do usuário de importância variada. Por isso, o XTP oferece um poderoso mecanismo de discriminação no nível de Transporte. Os usuários podem utilizar um campo de 32 bites que indica a prioridade da mensagem num espaço de 4 bilhões de possibilidades.
- Comunicações ponto a multi-ponto: o protocolo XTP oferece um mecanismo de comunicações de grupo semelhante ao do VMTP. Um transmissor pode endereçar um grupo de qualquer tamanho, substituindo *n* transmissões ponto a ponto por uma única transmissão ponto a multi-ponto. Como essa difusão é feita a nível de Transporte, ela é confiável fim-a-fim pois é submetida aos controles de fluxo, taxa e erro.
- Endereçamento: ao invés de introduzir outro esquema de endereçamento, o XTP pode operar com vários esquemas diferentes, incluindo endereços IP e ISO 8348.
- Roteamento: o mesmo pacote usado para estabelecimento de conexão estabelece o caminho, ou rota, da conexão. Posteriormente, pacotes especiais são utilizados para manutenção do caminho.

- Alinhamento: o protocolo XTP alinha seus campos em 4 octetos, usando campos de *offset* e comprimento para identificar o começo e o fim dos dados do usuário.

A Figura 3 mostra a arquitetura da máquina de estados do XTP. O processo RECEPTOR atua nos pacotes recebidos da rede e coloca os dados do usuário na fila de entrada correspondente. Estes dados serão entregues ao usário do XTP pelo processo LEITOR, que se comunica com a camada superior através de um Bloco de Controle. Um usuário do XTP que deseja enviar dados a um usuário remoto transfere os dados ao processo ESCRITOR (através de comandos do Bloco de Controle), que coloca os dados na fila de saída apropriada. O processo TRANSMISSOR entrega o conteúdo dessa fila à rede.

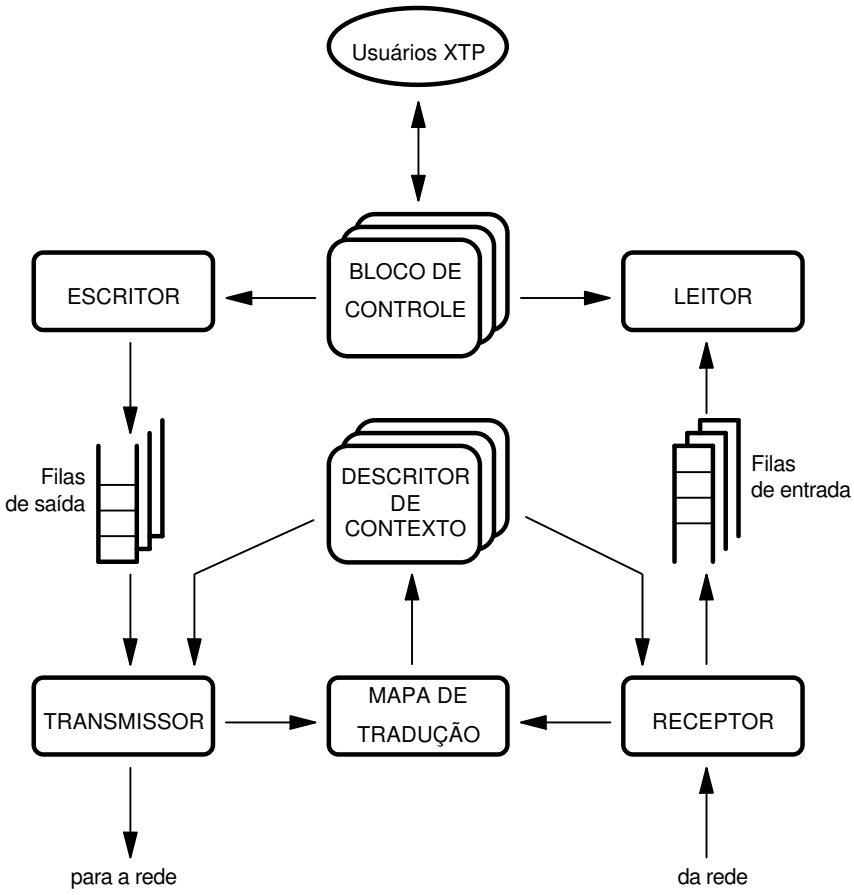

Figura 3. Arquitetura XTP.

Os Descritores de Contexto são as estruturas de dados que armazenam as informações de estado a respeito de cada conexão ativa, ou seja, são os descritores da conexão. A Tabela de Tradução mapeia os pacotes em seus Descritores de Contexto correspondentes. O processo TRANSMISSOR utiliza a Tabela de Tradução para acessar um Descritor de Contexto, onde realiza a atualização das variáveis de estado locais e coloca o conteúdo de algumas delas no pacote a ser transmitido. O processo RECEPTOR utiliza informações dos pacotes recebidos para indexar a Tabela de Tradução e portanto determinar o Descritor de Contexto que guarda informações relevantes àquele pacote.

Uma visão fim-a-fim da transferência de dados no XTP pode ser vista na Figura 4. Quando um usuário do XTP deseja se comunicar com um ou mais usuários remotos, ele estabelece uma conexão. Para uma dada conexão, as informações de estado armazenadas em uma estação são chamadas de contexto. Um contexto é também

chamado extremidade porque é o término do fluxo de dados entre os membros da conexão. Uma conexão é estabelecida quando uma extremidade iniciadora envia um pacote de etabelecimento de conexão para a estação remota, que cria uma extremidade correspondente para aquela conexão. O protocolo XTP fornece tanto conexões ponto a ponto quanto ponto a multi-ponto.

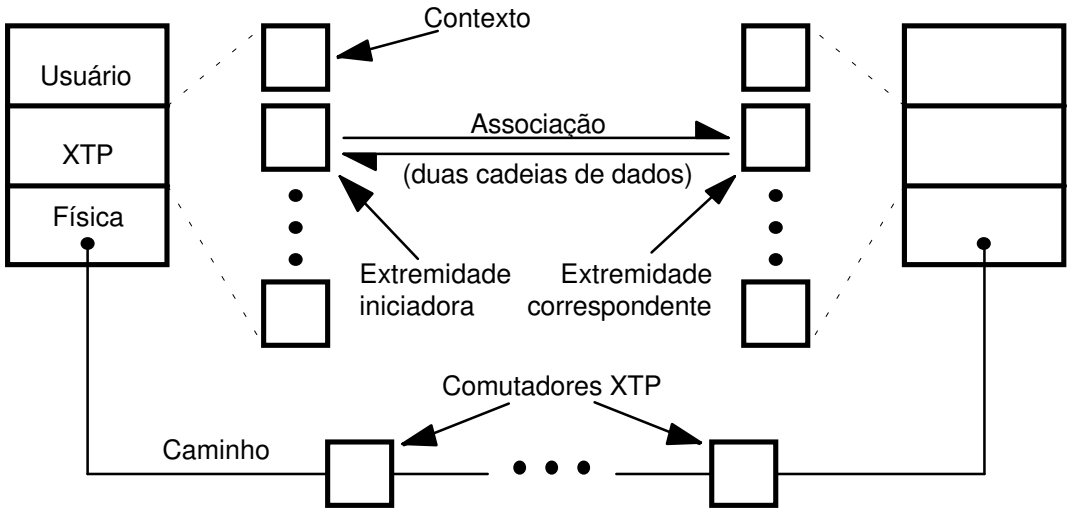

Figura 4. Modelo de comunicação entre entidades pares.

Uma conexão é composta por dois fluxos de dados distintos, um para cada direção, chamados cadeias de dados (a mesma idéia de *stream* do Datakit). Uma cadeia de dados é uma longa seqüência de dados ordenada, que é transferida de uma extremidade local a uma ou mais extremidades remotas. Extremidades de uma conexão não precisam necessariamente residir na mesma rede local. Uma conexão pode englobar vários *gateways* entre redes, que são os chamados comutadores. O conjunto ordenado de nós de uma extremidade da conexão a outra é chamado de caminho. Um pacote de estabelecimento de caminho deixa uma trilha de informações de roteamento em cada nó por onde passa. O protocolo XTP força o tráfego que flui no sentido reverso da cadeia de dados de uma conexão a percorrer os mesmos nós do sentido direto.

3 Conclusões

Neste artigo, foi apresentada uma visão geral de alguns protocolos projetados especificamente para aproveitar ao máximo os recursos oferecidos pelas novas redes de comunicação. É necessária, entretanto, uma comparação dos diferentes mecanismos utilizados por cada protocolo para atingir seus objetivos, de maneira a poder traçar as características gerais que um protocolo de Transporte deve ter para alcançar alto desempenho:

- **Estabelecimento e liberação de conexões:** observando-se os protocolos descritos neste artigo, pode-se concluir que, de uma maneira geral, em aplicações nas quais alta vazão é importante, o estabelecimento negociado de uma conexão (*handshake*) não introduz *overhead* significativo. Por outro lado, em aplicações mais interessadas em respostas rápidas, um mecanismo de pedido/resposta é mais adequado. Porém, isto nada mais é do que estabelecimento e liberação implícitos de conexão, onde informações de estado devem ser mantidas por um tempo às vezes inaceitavelmente grande em redes com longos atrasos. Há também o problema da disponibilidade de

recursos para manter os descritores de todas as conexões ativas, que pode se tornar grave em servidores que suportem várias conexões simultaneamente. Portanto, a melhor solução parece ser uma combinação dos dois métodos, como no XTP, onde o estabelecimento da conexão é implícito e a liberação é negociada.

- **Negociação de parâmetros:** é desejável que um protocolo seja flexível no sentido de inicialmente negociar e posteriormente modificar os parâmetros que forem necessários para adaptar o protocolo a diferentes ambientes. Todos os protocolos apresentados possuem esta funcionalidade.

- **Reconhecimentos:** quase todos os protocolos apresentados submetem o envio de reconhecimentos à recepção dos dados, a não ser no XTP, onde o transmissor solicita o envio de reconhecimentos. Entretanto, alguns autores [19] defendem que todas as informações de estado relevantes à conexão sejam trocadas periodicamente, independentemente de qualquer evento relacionado à conexão. Isto poderia ser feito em pacotes separados, aumentando o potencial de paralelismo do protocolo.

- **Formato dos pacotes:** as informações de identificação dos pacotes devem ser colocadas no cabeçalho, enquanto que nota-se uma tendência em se colocar o *checksum* depois dos dados, de tal forma que possa ser computado em paralelo com sua transmissão e recepção.

- **Controle de fluxo:** as mais modernas redes de comunicação, baseadas nas tecnologias RDSI-BL (Rede Digital de Serviços Integrados - Banda Larga) e ATM (*Asynchronous Transfer Mode*), implementam o controle de acesso ao meio utilizando taxas de transmissão média e de pico [20]. Neste contexto, o mecanismo de controle de fluxo mais apropriado é o controle de taxa. Através deste mecanismo, parâmetros *default* para a taxa de transmissão seriam obtidos diretamente da rede, evitando assim problemas de congestão.

- **Controle de erros:** apesar do *overhead* introduzido em sua computação, o mecanismo de *checksum* é o modo mais seguro de detecção de erros, e deve ser pelo menos opcional. Quanto à recuperação de erros, novamente a solução mais flexível adotada pelo XTP, que implementa tanto *go-back-n* quanto retransmissão seletiva (opcionais, dependendo da aplicação), parece ser a mais adequada, pois ambas as maneiras apresentam vantagens e desvantagens. A vantagem do *go-back-n* é que o protocolo fica mais simples. A desvantagem é que sua utilização pode gerar a retransmissão de dados corretamente recebidos, mas descartados. A retransmissão seletiva resolve este problema, apesar de forçar o receptor a ter recursos suficientes para armazenar e reseqüenciar os dados, o que pode ser um problema.

Finalmente, é necessário enfatizar que, ao mesmo tempo em que é importante tentar sempre desenvolver e melhorar os mecanismos do protocolo, é importante também tomar cuidado com a maneira como estes mecanismos são implementados, o que pode alterar significativamente o desempenho como um todo. Também deve merecer atenção especial o *overhead* introduzido pelo sistema operacional e pelos *drivers* dos dispositivos de entrada e saída, como as placas de rede. Daqui para a frente, é evidente que serão beneficiados os protocolos que puderem ser eficientemente implementados em VLSI, pois poderão realizar o processamento na mesma velocidade com que os dados são recebidos da rede.

Referências

[1] A. G. Fraser, "The Universal Receiver Protocol", *Proceedings of the IFIP Workshop on Protocols for High-Speed Networks*, Zurique, 9-11 de maio de 1989.

[2] A. G. Fraser e W. T. Marshall, "Data Transport in a Byte-Stream Network", *IEEE Journal of Selected Areas in Communication*, vol. 7, no. 7, pp. 1020-1033, setembro de 1989.

[3] G. Chesson, "Datakit Software Architecture", *Proceedings of the ICC*, pp. 20.2.1-20.2.5, 1979.

[4] International Organization for Standardization, "Information Processing Systems - Open Systems Interconnection - Basic Reference Model", *Draft International Standard 7498,* outubro de 1984.

[5] R. W. Watson, "Delta-t Protocol Specification", Lawrence Livermore Laboratory, 15 de abril de 1983.

[6] R. W. Watson, "The Delta-t Transport Protocol: Features and Experience", *Proceedings of the IFIP Workshop on Protocols for High-Speed Networks*, Zurique, 9-11 de maio de 1989.

[7] R. W. Watson, "The Delta-t Transport Protocol", *Proceedings of the 14th. Conference on Local Computer Networks*, Minneapolis, Minnesota, pp. 415-420, 10-12 de outubro de 1989.

[8] D. D. Clark, M. L. Lambert e L. Zhang, "NETBLT: A Bulk Data Transfer Protocol", Network Information Center RFC 998, SRI International, março de 1987.

[9] D. D. Clark, M. L. Lambert e L. Zhang, "NETBLT: A High Throughput Transport Protocol", *Proceedings of ACM SIGCOMM 87 Workshop: Frontiers in Computer Communications Technology*, Stowe, Vermont, pp. 353-359, agosto de 1987.

[10] D. R. Cheriton e C. L. Williamson, "VMTP as the Transport Layer for High-Performance Distributed Systems", *IEEE Communications Magazine*, vol. 27, no. 6, pp. 37-44, junho de 1989.

[11] D. R. Cheriton, "VMTP: Versatile Message Transaction Protocol, Protocol Specification", RFC 1045, Network Information Center, SRI International, fevereiro de 1988.

[12] D. R. Cheriton, "VMTP: A Transport Protocol for the Next Generation of Communication Systems", *Computer Communications Review*, vol. 16, no. 3, pp. 406-415, agosto de 1986.

[13] E. Nordmark e D. R. Cheriton, "Experiences from VMTP: How to achieve low response time", *Proceedings of the IFIP Workshop on Protocols for High-Speed Networks*, Zurique, 9-11 de maio de 1989.

[14] Protocol Engines Inc., "Xpress Transfer Protocol Definition", Revisão 3.6, janeiro de 1992.

[15] G. Chesson e L. Green, "XTP-Protocol Engine VLSI for real-time LANs", *Proceedings of the EFOC/LAN*, Amsterdam, pp. 1-4, julho de 1988.

[16] G. Chesson, "XTP PE design considerations", *Proceedings of the IFIP Workshop on Protocols for High-Speed Networks*, Zurique, 9-11 de maio de 1989.

[17] J. Postel, "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, USC/Information Sciences Institute, setembro de 1981.

[18] International Organization for Standardization, "Information Processing Systems - Open Systems Interconnection - Transport Protocol Specification", *Draft International Standard 8073*, julho de 1986.

[19] A. N. Netravali, W. D. Roome e K. Sabnani, “Design and Implementation of a High-Speed Transport Protocol”, *IEEE Transactions on Communications*, vol. 38, no. 11, pp. 2010-2023, novembro de 1990.
[20] M. de Prycker, “Asynchronous Transfer Mode - Solution for Broadband ISDN”, Ed. Ellis Horwood, 1991.