Análise de Mecanismos para Protocolos de Alto Desempenho

Marcelo Dias Nunes
email:bill@coe.ufrj.br

Otto C. M. B. Duarte
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

Resumo

Nos últimos anos, houve um aumento significativo da banda passante nos meios de comunicação, tranferindo a limitação de desempenho para o processamento da informação. As novas redes de comunicação deverão veicular aplicações com diferentes requisitos. Os novos protocolos deverão ser bastante flexíveis, permitir alta velocidade e comunicações multidestinatárias, onde a utilização de mecanismos de alto desempenho é de fundamental importância. Assim, este artigo apresenta uma análise e uma comparação dos mecanismos de gerenciamento de conexões, reconhecimentos, controle de fluxo e controle de erros, visando traçar as linhas gerais de um protocolo de alto desempenho adequado à atual multiplicidade de serviços.

Abstract

In the latest years, there has been a significant increase on the communication media bandwidth, shifting performance limitation to information processing. New communication networks shall provide applications with different requirements. New protocols must be very flexible, allow high speed and multipoint communications, wherein high performance mechanisms' utilization is of fundamental importance. Thus, this paper presents an analysis and a comparison of connection management, acknowledgment, flow control and error control mechanisms, in view of defining the general outlines of a high performance protocol suitable for the present multiplicity of services.

1 Introdução

Nos anos 60, a banda-passante dos meios de comunicação era um recurso caro e escasso em redes de longa distância. Especificamente, taxas de transmissão de centenas, no máximo milhares, de bits por segundo eram disponíveis. Além disso, o serviço oferecido pela rede era de baixa qualidade, com uma grande freqüência de erros de transmissão.

Nos últimos anos, com o desenvolvimento das tecnologias VLSI e de transmissão por fibra ótica, a velocidade das redes de comunicação aumentou significativamente, atingindo 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 gigabits por segundo. O fator limitante tranferiu-se da banda-

passante para o processamento do protocolo, que impede uma aplicação de utilizar uma fração razoável da banda disponível.

Além disso, as futuras Redes Digitais de Serviços Integrados em banda larga deverão oferecer serviços que suportam uma vasta gama de aplicações: transferência de arquivos, transações cliente/servidor, datagrama, transferências multidestinatárias e aplicações de tempo real.

Para se atingir alto desempenho, várias estratégias têm sido propostas, tais como: arquiteturas de implementação eficazes, utilização de processamento paralelo [1,2,3] e implementação em *hardware* do conjunto de protocolos.

Uma das etapas fundamentais para se definir um protocolo que atenda às diferentes características das diversas aplicações é o estudo dos mecanismos de comunicação usados nos protocolos de alto desempenho citados na literatura especializada.

Neste artigo, será feita uma análise de diferentes mecanismos necessários para o suporte dos seguintes serviços:

- gerenciamento de conexões, incluindo os métodos de abertura, manutenção e liberação;
- reconhecimentos, usados pelo receptor para informar ao transmissor o sucesso ou fracasso da recepção correta dos dados;
- controle de fluxo, para compatibilizar as taxas de transmissão e recepção e, com isso, evitar congestionamentos;
- controle de erros, incluindo detecção e recuperação de dados que possam ser perdidos por erros de transmissão e *overrun*.

Na Seção 2 são analisados, em linhas gerais, os mecanismos utilizados por cinco protocolos, alguns de transporte e outros que possuem esta funcionalidade: os protocolos Datakit, Delta-t, NETBLT, VMTP e XTP. Todos são protocolos que, de uma forma ou de outra, procuram simplificar o processamento para atingir alto desempenho. Porém, cada um é mais adequado a uma ou outra aplicação particular. Na seção 3, são comparados os mecanismos descritos na seção anterior, destacando suas vantagens e desvantagens. Conclusões são apresentadas na Seção 4, numa tentativa de delinear o perfil de um protocolo de transferência de propósito geral, visando alto desempenho.

2 Alguns protocolos de alto desempenho

2.1 Datakit

O protocolo Datakit [4,5,6] foi desenvolvido na tentativa de tornar uma rede receptiva às diferentes necessidades dos usuários, ou seja, dotá-la da capacidade de atender a diversos serviços (computação interativa, transferência de arquivos, transações cliente/servidor, transmissão de voz, etc.). Sua idéia básica visa uma arquitetura flexível, simplificando ao máximo o processamento para atingir altas vazões. Os projetistas decidiram adotar o octeto como unidade indivisível de informação e a cadeia de octetos como paradigma de comunicações, inspirado no sistema de arquivos do UNIX, em que cada arquivo é representado por uma cadeia de octetos. À unidade básica de informação é adicionado 1 bit para distinguir um octeto de dado de um octeto de controle.

A funcionalidade do protocolo Datakit pode ser mapeada nas funções das camadas física, enlace e transporte da arquitetura OSI da ISO [7]. No nível de transporte do Datakit, os projetistas desenvolveram um protocolo "receptor universal" (URP — *Universal Receiver Protocol*) a partir da observação de que o desempenho geral do sistema é comprometido pelo processamento no receptor. Padronizando-se o protocolo do receptor e permitindo-se que o transmissor seja implementado de acordo com as necessidades de cada aplicação, garante-se a interoperabilidade entre todas as estações da rede, desde que cada transmissor seja capaz de se comunicar com o URP. Toda a responsabilidade da conexão é transferida para o transmissor, sendo o receptor apenas uma máquina **passiva** que responde a comandos do transmissor. Esses comandos (e suas respectivas respostas) são octetos de controle.

A arquitetura do URP consiste basicamente em um processo que coloca cada octeto recebido numa fila FIFO (dividida em uma área de armazenamento que contém os dados não checados e uma fila principal que contém os dados recebidos corretamente) e um processo que retira os dados da fila e os entrega ao usuário (Figura 1).

Figura 1: A estrutura do Receptor Universal.

Conexões: Numa rede interligada pelo Datakit, as conexões, ou circuitos virtuais, são gerenciadas por estações dedicadas, utilizando *hanshake* com duas mensagens nas etapas de estabelecimento e liberação (*two-way handshake*). No sentido de simplificar o processamento, a especificação do Datakit não diz nada a respeito do gerenciamento de conexões. Um procedimento específico deve ser desenvolvido para esta finalidade, caracterizando estabelecimento e liberação de conexão "fora da banda". A camada enlace do Datakit é responsável pela multiplexação de diferentes conexões de transporte em uma única conexão de enlace, agrupando os octetos de uma mesma conexão em quadros com um cabeçalho identificando a conexão. Um par transmissor/receptor suporta apenas conexões *simplex*, sendo necessários dois pares para configurar uma transmissão de dados *full duplex*. Porém, os dados e comandos do transmissor local podem ser multiplexados com as respostas do receptor local ao transmissor remoto na mesma conexão *simplex*, uma vez que os códigos de comandos e respostas são diferentes.

Reconhecimentos: No Datakit, os reconhecimentos são solicitados pelo transmissor, através de comandos específicos.

Controle de fluxo: O mecanismo de controle de fluxo empregado no Datakit é o de janelas de transmissão.

Controle de erros: Há dois modos de operação para o URP, **modo bloco** e **modo caracter**. No **modo bloco**, os octetos da mensagem são agrupados em blocos seguidos de uma cauda que contém o número de octetos do bloco (sem considerar o tamanho da cauda). O controle de erros é feito comparando-se a contagem dos octetos de dados com o campo de comprimento. Todo erro se manifesta à camada transporte como informação perdida uma vez que a camada enlace descarta os octetos corrompidos. Há ainda dois tipos de blocos, **blocos de mensagem** e **blocos**

de cadeia de dados. Nos dois tipos, blocos incorretamente recebidos são descartados. No primeiro tipo, após o descarte, nenhum novo bloco é aceito até que uma retransmissão (por *go-back-n*) do bloco perdido seja corretamente recebida. No segundo tipo, novos blocos continuam sendo aceitos após o descarte de um bloco corrompido. Já no **modo caracter**, espera-se que o transmissor envie rajadas de dados seguidas de um número de seqüência. A área de armazenamento não é usada e todos os dados que entram na fila são imediatamente transferidos ao usuário, não havendo portanto controle de erros.

Observa-se, portanto, que as maiores contribuições do Datakit para o desenvolvimento de protocolos de alto desempenho foram: submeter todo o controle da comunicação ao transmissor, usar uma unidade de informação mínima baseada no octeto e, juntamente com a possibilidade do não controle de erros no modo caracter, viabilizar o empacotamento de voz para aplicações de tempo real. O transmissor pode então implementar a política mais adequada à aplicação usuária. Contudo, apesar de toda a simplificação do processamento do receptor, o uso de *bytes* de 9 bits impõe um *overhead* significativo no desempenho do sistema. Além do tratamento ser realizado *byte* a *byte*, há problemas de alinhamento decorrentes do uso de PDUs (*Protocol Data Units*) de 9 bits. A rigor, não passam de mensagens com um campo de informação de 8 bits e um cabeçalho de 1 bit. Porém, um dos objetivos dos projetistas do Datakit era a implementação em *hardware* deste protocolo. Neste caso, o *overhead* introduzido pelo processamento em *software* desaparece.

2.2 Delta-t

O protocolo Delta-t [8,9,10] foi desenvolvido para integrar desde micro a supercomputadores em redes heterogêneas. É um protocolo de propósito geral que suporta tanto transações (pedido/resposta) quanto transferências de grandes quantidades de dados em alta velocidade, numa rede onde possam ocorrer perdas de dados, duplicações e/ou dados fora de seqüência. A funcionalidade do Delta-t é a de um protocolo de rede não orientado a conexão e de um protocolo de transporte. Uma nova política de gerenciamento de conexões foi projetada, totalmente baseada em temporizadores, uma vez que os protocolos tradicionais, como o TCP [11] e o OSI TP4 [12], não permitiam uma implementação eficiente de transações, devido ao *overhead* introduzido pelo *handshake* de estabelecimento de conexões.

Conexões: A principal característica do Delta-t e a maior contribuição de seus projetistas para o desenvolvimento de protocolos de alto desempenho foi a exaustiva análise realizada sobre os mecanismos de gerenciamento de conexões, que levou à decisão pelo gerenciamento de conexões por temporização. Para chegar a esta conclusão, foram analisados dois outros mecanismos básicos de gerenciamento de conexões: *handshake* e identificadores de conexão únicos. O mecanismo de *handshake*, entretanto, necessita de temporizadores nas etapas de estabelecimento e liberação de conexões para evitar algumas situações patológicas, como mensagens duplicadas de uma conexão previamente fechada causarem a abertura de uma nova conexão, ou a conexão ser fechada antes que todos os dados retransmitidos sejam reconhecidos. O uso de identificadores de conexão únicos traz o problema de se garantir identificadores únicos em toda a rede, e assegurar que esses identificadores não sejam reutilizados por um intervalo de tempo (novamente se faz necessário o uso de temporizadores) suficiente para permitir a expiração de todas as mensagens duplicadas. Por isso, os autores decidiram adotar o mecanismo de temporização para assegurar o que eles mesmo chamam de "gerenciamento de conexões livre de riscos". Na nomenclatura Delta-t, o

termo conexão define o intervalo de tempo durante o qual informações de estado são mantidas em cada extremidade comunicante. A abertura de uma conexão consiste em se alocar este estado confiavelmente e a liberação consiste na dealocação deste estado confiavelmente e sem ambigüidades. Num estilo que ficou conhecido como **abertura de conexão implícita**, a primeira mensagem da conexão automaticamente a estabelece, alocando informações de estado. A partir daí, temporizadores são mantidos no transmissor e no receptor durante a transferência de dados. O temporizador do receptor é atualizado cada vez que um novo identificador é aceito ou a conexão é fechada. Ao final da transferência de dados, as informações de estado são mantidas até que todas as retransmissões sejam corretamente recebidas e todas as duplicatas sejam reconhecidas. O temporizador do transmissor garante que, no caso da queda de algum nó da rede, os números de seqüência das mensagens de dados e de reconhecimento não sejam reutilizados até que tenham expirado. Para que isso seja possível, no cabeçalho de todas as mensagens existe um campo de tempo-de-vida que é decrementado durante as etapas de roteamento (em cada nó), retransmissão e reconhecimento. Este procedimento, entretanto, onera o processamento nos nós da rede, obrigados a implementar mecanismos para capturar as mensagens, decrementar seus campos de tempo-de-vida e devolvê-los para a rede. Por fim, se não forem explicitamente terminadas, as conexões são automaticamente liberadas e as informações de estado dealocadas quando estourarem os temporizadores do transmissor e do receptor, caracterizando a **liberação de conexão implícita**. As conexões são identificadas por uma **tripla** que consiste dos endereços da porta destino (números de 64 bits), da porta origem e de um número de conexão.

Reconhecimentos: No Delta-t, a geração de reconhecimentos é vinculada à recepção dos dados, portanto independente do transmissor.

Controle de fluxo: O mecanismo de controle de fluxo utilizado é o de janelas de transmissão.

Controle de erros: Os dados são retransmitidos com base em informações obtidas do receptor. O mecanismo de retransmissões adotado pelo Delta-t é o *go-back-n*.

2.3 NETBLT

O protocolo NETBLT (*NETwork BLock Transfer*) [13,14] é um protocolo de transporte projetado para atingir altas velocidades em aplicações de transferência de grandes volumes de dados. Por isso, uma minuciosa análise dos mecanismos de controle de fluxo, controle de erros e da influência dos atrasos na transmissão foi realizada para se encontrar a melhor solução que garantisse alto desempenho. A operação básica do NETBLT consiste em se abrir uma conexão entre duas entidades, transferir dados numa série de blocos (*buffers*) numerados e fechar a conexão. Cada *buffer* é composto de um grande número de mensagens.

Conexões: As conexões no NETBLT são estabelecidas e terminadas através de *handshake* com duas mensagens (*two-way handshake*).

Reconhecimentos: Existe uma mensagem de reconhecimento positivo, indicando que todas as mensagens de um *buffer* foram corretamente recebidas, e uma mensagem de reconhecimento negativo, contendo uma lista das mensagens perdidas.

Controle de fluxo: Com o propósito de aumentar a eficiência da comunicação em termos de vazão, o NETBLT desacopla os mecanismos de controle de fluxo e de erros, de tal forma que a detecção de mensagens perdidas não esteja amarrada a como e quando o controle de fluxo é em-

pregado. Portanto, os projetistas do NETBLT introduziram um novo mecanismo de controle de fluxo, o **controle de fluxo por taxa**. A rigor, este não é o único mecanismo de controle de fluxo disponível no NETBLT. Quando é estabelecida uma conexão entre o transmissor e o receptor, o NETBLT confirma que ambos possuem *buffers* de mesma capacidade. Uma vez que a quantidade de dados a ser transmitida pode ser muito grande, a aplicação fornece os dados em *buffers* com essa capacidade. À aplicação é então permitido controlar efetivamente o fluxo, através do fornecimento (ou não) de um novo *buffer*, de acordo com uma política particular e transparente ao NETBLT. Mas uma vez que os *buffers* podem ser grandes, é necessário o controle do fluxo durante a transmissão de um *buffer*. E o fluxo é controlado pelo pré-estabelacimento da taxa de transmissão. O transmissor envia então uma rajada de mensagens num determinado intervalo de tempo. Durante a etapa de estabelecimento de conexão, o tamanho da rajada e a taxa são negociadas, fixando indiretamente o temporizador da rajada. No decorrer da transmissão de um *buffer*, a taxa não pode ser alterada, só podendo ser renegociada ao final.

Controle de erros: Um problema que ocorre com mecanismos de retransmissão é a determinação do valor dos temporizadores de retransmissão, independente da maneira como são implementados. Se o temporizador for muito longo, ao se atingir o limite superior da janela de transmissão, corre-se o risco de se esperar pelo estouro do temporizador, caso alguma mensagem seja perdida (nenhum ACK será enviado). Por outro lado, se o temporizador for curto demais, ou mais especificamente, menor que o tempo levado pelo ACK para chegar, corre-se o risco de se disparar retransmissões desnecessárias, congestionando a rede e o receptor. Logo, percebe-se que o valor dos temporizadores está intimamente relacionado com o *round trip delay* (RTD), ou tempo de ida e volta, que na prática é aleatoriamente variável. A pré-negociação da taxa de transmissão facilita em muitos aspectos a transmissão de dados. Assim que a primeira mensagem de dados de um determinado *buffer* chega no receptor, é disparado um temporizador cujo valor é facilmente calculado, uma vez que a taxa de transmissão e o tamanho do *buffer* são conhecidos. Se o *buffer* for completa e corretamente recebido dentro do tempo esperado, o temporizador é interrompido (neste caso, a janela de transmissão é de um *buffer*, mas para manter a transmissão contínua, esta janela pode ser fixada em dois ou mais *buffers*, de acordo com as disponibilidades de recursos da estação). Porém, se o temporizador estourar, o receptor solicita retransmissão. Os dados a serem retransmitidos são colocados no mesmo *buffer* com os novos dados, sendo portanto liberados à mesma taxa, evitando sobrecargas na rede. É importante destacar aqui que o temporizador de retransmissão se encontra no **receptor**, numa metodologia diferente da habitual. Isso advém da observação pelos projetistas do NETBLT de que apenas o receptor sabe exatamente que mensagens foram recebidas, eliminando assim retransmissões desnecessárias.

Resumindo, pode-se dizer que a principal contribuição do NETBLT para o projeto de protocolos de alta velocidade é no campo do controle de fluxo, com a introdução do controle de fluxo por taxa. Entretanto, isto exige uma rede que consiga interagir com o NETBLT para a determinação do valor inicial da taxa de transmissão, bem como permitir futuras alterações desse valor inicial, fazendo com que a taxa reflita sempre as condições operacionais atuais da rede. Por isso, não podem haver mudanças bruscas nas condições de carga da rede, de tal forma que a informação obtida dos *gateways* seja útil ao chegar. Isso tudo implica em modificações no processamento de nós e *gateways*, o que pode não ser viável.

2.4 VMTP

O VMTP (*Versatile Message Transaction Protocol*) [15,16,17,18] foi projetado visando suprir uma deficiência dos protocolos de transporte convencionais, que ofereciam pouco (ou nenhum) suporte a serviços típicos de sistemas distribuídos, como acesso a páginas de arquivos, chamadas remotas de procedimentos, datagramas em tempo real e comunicações ponto-a-multiponto. O VMTP é um protocolo de transporte basicamente orientado a transações (pedidos e respostas). Na nomenclatura VMTP, a entidade que solicita um serviço através de um pedido é chamada **cliente**, enquanto a entidade que atende o pedido através de uma resposta é chamada **servidora**.

Conexões: O VMTP também usa o mecanismo de conexão implícita usado no Delta-t. Desta forma, uma transação de mensagens básica no VMTP consiste em um cliente enviando uma mensagem de pedido a uma entidade servidora e recebendo de volta uma mensagem de resposta. 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 se o temporizador a ele associado estourar (desconexão implícita). De maneira semelhante, cada cliente exige apenas um descritor, independentemente do número de transações em que esteja envolvido. O protocolo VMTP oferece ainda três variações da transação básica. 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. O pedido é retransmitido até que pelo menos uma resposta seja recebida (ou até que o temporizador estoure). Um grupo de entidades pode ser identificado por um único identificador de entidade, mesmo que elas estejam distribuídas por várias estações. O VMTP é portanto o primeiro a introduzir suporte a conexões ponto-a-multiponto (*multicast*). Opcionalmente, o VMTP permite que o servidor envie uma "resposta de grupo" ao cliente e aos membros do grupo destino de servidores ao qual o pedido foi previamente enviado. Na segunda variante, o cliente envia um pedido como um **datagrama** com uma indicação de que nenhuma resposta é esperada. Na última variante, a mensagem de pedido pode ser **expedida** (*forwarded*) para outro servidor, sendo que caberá àquele servidor responder diretamente ao cliente. Desta forma, o cliente não precisa saber que uma mensagem teve que sofrer novo roteamento.

Reconhecimentos: A resposta do servidor funciona como reconhecimento para o cliente. 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. O próximo pedido do cliente ou o estouro do temporizador funcionam como reconhecimento para o servidor. É importante frisar, porém, que também existem reconhecimentos explícitos de pedidos e respostas, enviados como datagramas. Especificamente, uma mensagem *NotifyClient* é enviada pelo servidor para reconhecer um pedido, enquanto um *NotifyServer* é enviado pelo cliente para reconhecer uma resposta. O cliente só pode ter uma transação de mensagens não reconhecida de cada vez, não importando a quantidade de dados transferida. No modo multiponto, o VMTP transfere para a aplicação a decisão de implementar ou não *multicast* confiável. Uma aplicação que necessite de reconhecimentos positivos pode ser implementada de maneira a aceitar todas as respostas recebidas pelo VMTP. No modo datagrama, o pedido pode ser enviado confiavelmente ou não (com ou sem reconhecimento).

Controle de fluxo: Assim como no NETBLT, o VMTP utiliza controle de fluxo por taxa, mas o mecanismo adotado é diferente: "espaços", ou intervalos (*gaps*), são introduzidos entre as mensagens para reduzir a taxa de transmissão. Clientes e servidores podem explicitamente comunicar os intervalos desejados.

Controle de erros: Também como no NETBLT, o VMTP utiliza retransmissão seletiva. Os pedidos de retransmissão seletiva auxiliam o transmissor a fazer ajustes no controle de fluxo, reduzindo o intervalo quando não ocorrem perdas de mensagens, 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*.

Endereçamento: Até o surgimento do 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.

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 protocolo XTP (*Xpress Transfer Protocol*) [19,20,21].

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 fato de que apenas o usuário conhece suficientemente a aplicação para melhor otimizar os parâmetros da comunicação. O Datakit já preconizava esta abordagem. Aliás, o XTP incorpora características de todos os protocolos analisados anteriormente neste artigo, além dos protocolos TCP e OSI TP4. Entretanto, não é exatamente um protocolo de transporte, e sim de **transferência**, pois combina também funcionalidades de rede, como roteamento. Mas apesar de toda sua potencialidade, ainda é baseado numa máquina de estados finita, visando implementação em *hardware*.

Conexões: O XTP consegue confiabilidade com a troca de apenas duas mensagens. Este estabelecimento implícito de conexões pode porém ser substituído pelo *handshake* habitual. A mesma mensagem usada para estabelecimento de conexão (FIRST) estabelece o caminho, ou rota, da conexão. Posteriormente, mensagens especiais são utilizadas para manutenção do caminho (ROUTE). Eventualmente, pode ser necessário enviar informações a respeito da conexão sem encaixá-las na conexão propriamente dita. O XTP permite que cada mensagem carregue oito octe-

tos de dados "marcados". Estes dados são entregues ao usuário final, mas não são interpretados pelo XTP. Conexões ponto-a-multiponto também são oferecidas, como no VMTP. Um transmissor pode endereçar um grupo de qualquer tamanho, substituindo *n* transmissões ponto a ponto por uma única transmissão ponto-a-multiponto (*multicast*). Como essa difusão é feita a nível de transporte, ela pode ser opcionalmente submetida aos controles de fluxo, taxa e erro. Além disso, o XTP permite que usuários se conectem dinamicamente a um grupo. Se um usuário desejar se unir a uma conexão multiponto em andamento (na condição de receptor), ele envia uma mensagem especial com destino ao grupo. O transmissor, ao receber esta mensagem, retorna àquele usuário informações sobre a conexão que permitam a ele participar da comunicação. Quando um usuário quiser se desligar, basta parar de participar da transferência de dados, não sendo necessário notificar o transmissor. Conexões multiponto são estabelecidas implicitamente pelo transmissor com uma mensagem FIRST endereçada ao grupo. No decorrer da conexão, o transmissor pode solicitar, através de mensagens CNTL, informações sobre o *status* do grupo de receptores. Para evitar que o transmissor deva conhecer o número de receptores, o XTP sugere a implementação de algumas técnicas (que não fazem parte da especificação), como *damping*, *slotting* e o algoritmo *bucket*. O fechamento de uma conexão oferece várias opções, como fechamento bidirecional (*two-way handshake*), unidirecional e abortos. As conexões ponto-a-multiponto são explicitamente terminadas pelo transmissor.

Reconhecimentos: os pedidos de reconhecimento são feitos explicitamente pelo transmissor, através de bits especiais (SREQ ou DREQ) no cabeçalho de mensagens de dados (DATA) ou controle (CNTL).

Controle de fluxo: O XTP utiliza janelas de transmissão de 32 bits, ou seja, permite até 4 bilhões de octetos não reconhecidos em cada conexão (no XTP, os octetos de dados, e não as mensagens, são seqüencialmente numeradas). Poderá passar facilmente para números de seqüência de 64 bits. O controle de taxa atua 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.

Controle de erros: Algumas aplicações não exigem a retransmissão de dados perdidos. No XTP, o esquema de retransmissões pode ser desligado. O XTP implementa tanto *go-back-n* como retransmissão seletiva, em que o receptor fornece um vetor indicando o número de seqüência abaixo do qual todos os octetos de dados foram corretamente recebidos mas ainda não entregues ao usuário (*rseq*), bem como as seqüências de informação corretamente recebidas (*spans*), de tal forma que o transmissor possa calcular e retransmitir apenas as lacunas (*gaps*). Quanto à detecção de erros, o cabeçalho de uma mensagem XTP contém um *checksum* próprio, enquanto o *checksum* dos dados 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.

Endereçamento: ao invés de introduzir um novo esquema de endereçamento, o XTP pode operar com vários esquemas existentes, incluindo endereços IP e ISO 8348.

O XTP oferece ainda a possibilidade de fixar prioridades, pois um problema com protocolos de transporte convencionais é que eles não são muito receptivos a dados do usuário de importân-

cia variada. Por isso, o XTP oferece um poderoso mecanismo de discriminação no nível de transporte. Os usuários podem utilizar um campo de 32 bits que indica a prioridade da mensagem num espaço de 4 bilhões de possibilidades. Além disso, por uma questão de eficiência, o XTP fixa o comprimento do cabeçalho e da cauda e a posição de todos os *flags* no cabeçalho, alinhando seus campos em 4 octetos. Campos de *offset* e comprimento para identificar o começo e o fim dos dados do usuário são utilizados.

3 Comparação dos mecanismos

Nesta seção, é feita uma comparação dos mecanismos utilizados pelos protocolos vistos na seção anterior, apresentando suas vantagens e desvantagens:

Conexões: De uma maneira geral, em aplicações de transferência de grandes volumes de dados, o *handshake* de conexão e desconexão não introduz *overhead* significativo. Por outro lado, em aplicações mais interessadas em respostas rápidas, um mecanismo de transação, com estabelecimento e liberação implícitos de conexão, é mais adequado. Para isso, contudo, é necessário que as estações disponham de recursos suficientes para manter os descritores de todas as conexões ativas pelo tempo necessário.

Reconhecimentos: Quase todos os protocolos apresentados submetem o envio de reconhecimentos à recepção dos dados, a não ser no XTP e no Datakit, onde é o transmissor quem solicita o envio de reconhecimentos. Com isso, o envio de ACKs fica desvinculado de qualquer evento específico no receptor, acelerando seu processamento, que é um fator crítico no desempenho global. Entretanto, alguns autores [22] defendem que todas as informações de estado relevantes à conexão sejam trocadas periodicamente. Isto poderia ser realizado por processos independentes com mensagens especiais, aumentando o potencial de paralelismo do protocolo.

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 [23]. 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 congestionamento. Além do mais, janelas de transmissão não controlam realmente o fluxo, mas apenas um parâmetro (o número de mensagens não reconhecidas) que está indiretamente relacionado ao fluxo.

Controle de erros: Apesar do *overhead* introduzido em sua computação, o mecanismo de *checksum* ainda é o modo mais seguro de detecção de erros, e deve ser implementado para uso pelo menos opcional. Quanto à recuperação de erros, a vantagem do *go-back-n* é que o processamento fica mais simples. A desvantagem é que sua utilização pode gerar a retransmissão de dados corretamente recebidos, mas descartados. Particularmente num ambiente multiponto, os receptores devem estar preparados para tratar duplicatas, caso as retransmissões sejam feitas para todo o grupo. A retransmissão seletiva evita retransmissões desnecessárias, apesar de forçar o receptor a ter recursos suficientes para armazenar e reseqüenciar os dados.

Negociação de parâmetros: É indispensável que um protocolo seja flexível no sentido de inicialmente negociar e posteriormente modificar os parâmetros que regem a comunicação. Se o usuário não desejar estabelecer algum parâmetro, o protocolo deve ser capaz de oferecer um valor

default. Isso implica, como já foi discutido anteriormente, em modificações no processamento dos nós da rede subjacente. Nenhum protocolo até hoje especificou exatamente a forma de negociação de parâmetros de qualidade de serviço, sendo este atualmente um vasto campo de pesquisa.

Formato das PDUs: As informações de identificação das PDUs 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, no caso de uma implementação em *hardware*. É indiscutível que os campos devem ter tamanho fixo, alinhados preferencialmente por 4 ou 8 octetos, para aproveitar o potencial das máquinas de 32 e 64 bits, bem como acelerar o processamento de decodificação dos campos.

4 Conclusões

Neste artigo, foram analisados mecanismos de comunicação de alguns protocolos de alto desempenho. Cada um deles, entretanto, é destinado a uma aplicação, ou classe de aplicações, particular. Para atender as necessidades emergentes das redes atuais de larga banda-passante, um protocolo não pode visar apenas alta velocidade, mas deve ser capaz de se adaptar e suportar a multiplicidade de aplicações exigidas pelos usuários, sejam elas transferência de arquivos ou transações, orientadas a conexão ou datagrama, comunicações ponto-a-ponto ou multidestinatárias. Portanto, os novos protocolos devem ser completamente configuráveis. Isto porém implica em, ao invés de se decidir pela implementação de um ou outro mecanismo de gerenciamento de conexões, controle de erros e controle de fluxo, implementar-se a princípio todos os mecanismos possíveis, uma vez que cada um é destinado a diferentes aplicações.

Com base nos protocolos analisados neste artigo, já se pode então traçar, em linhas gerais, o perfil de um protocolo de transferência de alto desempenho, que atenda a demanda das novas redes e serviços de comunicação.

O gerenciamento de conexões mais flexível e adaptável à maioria dos casos parece ser uma mistura dos mecanismos de temporização com *handshake*. Não é necessário mais do que uma mensagem para estabelecer uma conexão. Esta mensagem já pode conter os primeiros octetos de informação, se não todos, no caso de, por exemplo, uma chamada remota de procedimento. A manutenção de temporizadores associados aos descritores da conexão durante sua vida útil para desconexão automática em caso de inatividade é um recurso dispendioso, além do problema da determinação do valor mais apropriado para este temporizador. Um temporizador muito longo pode causar uma desnecessária retenção de recursos. Logo, uma proposta seria a desconexão negociada entre transmissor e receptor, com a utilização de um temporizador de desconexão, para garantir a dealocação confiável dos descritores e impedir que retransmissões ou duplicatas dêem origem a uma nova conexão.

O mecanismo de controle de fluxo por taxa é teoricamente melhor que o controle por janelas de transmissão, porque reflete com maior fidelidade as condições de tráfego da rede subjacente. Mas para que o controle por taxa seja efetivo, é necessário que a rede subjacente interaja de alguma maneira com o protocolo de transferência, fornecendo parâmetros atualizados, em tempo hábil, para a determinação da taxa de transmissão.

Quanto ao controle de erros, ambos *go-back-n* e retransmissão seletiva devem ser implementados para permitir a opção por um dos dois mecanismos, de acordo com a aplicação. Um misto das duas técnicas, em que retransmissão seletiva seria usada na primeira retransmissão de uma mensagem e *go-back-n* seria usado nas retransmissões seguintes (da mesma mensagem) [24], também seria interessante, pois a probabilidade da mensagem chegar com erro pela segunda vez é muito pequena.

Finalmente, para atender aos serviços de teleconferência e trabalhos cooperativos é imprescindível que o protocolo suporte conexões ponto-a-multiponto e multiponto-a-multiponto. Estes tipos de conexões poderão não ser completamente confiáveis, devido ao grande número de participantes de um grupo, que podem inclusive estar geograficamente dispersos. Uma conexão ponto-a-multiponto confiável implicaria em um tráfego reverso que poderia sobrecarregar a rede, além do fato de que diferentes receptores têm diferentes capacidades de processamento, e a comunicação não pode ser penalizada pelo processamento lento de um dos participantes. Dos protocolos analisados, somente o VMTP e o XTP suportam conexões ponto-a-multiponto. Entretanto, comunicações de conferência não são previstas em nenhum dos protocolos estudados. Com esta finalidade, um mecanismo de **fichas** deve ser implementado, onde cada membro que deseje transmitir dados deve solicitar a ficha para si, não podendo ficar com ela por mais do que um intervalo de tempo pré-estabelecido.

Referências

[1] O. G. Koufopavlou, A. N. Tantawy and M. Zitterbart, "Analysis of TCP/IP for High Performance Parallel Implementations", *Proc. 17th Conference on Local Computer Networks*, Minneapolis, Minesota, Sept. 1992, pp. 576-585.

[2] M. Zitterbart, "Parallel Protocol Implementation for High Speed Networks", in *Proc. SBT/IEEE Int. Telecommun. Symp.*, Rio de Janeiro, Brazil, Sept. 1990, pp. 260-264.

[3] C. Papadopoulos, G. M. Parulkar, "Experimental Evaluation of SunOS IPC and TCP/IP Protocol Implementation", *IEEE ACM Trans. Networking*, vol.1, no. 2, pp. 199-216, Apr. 1993.

[4] 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.

[5] 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.

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

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

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

[9] 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.

[10] 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.

[11] J. Postel, “Transmission Control Protocol - DARPA Internet Program Protocol Specification”, RFC 793, USC/Information Sciences Institute, setembro de 1981.
[12] International Organization for Standardization, “Information Processing Systems - Open Systems Interconnection - Transport Protocol Specification”, *Draft International Standard 8073*, julho de 1986.
[13] 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.
[14] 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.
[15] 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.
[16] D. R. Cheriton, “VMTP: Versatile Message Transaction Protocol, Protocol Specification”, RFC 1045, Network Information Center, SRI International, fevereiro de 1988.
[17] 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.
[18] 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.
[19] Protocol Engines Inc., “Xpress Transfer Protocol Definition”, Revisão 3.6, janeiro de 1992.
[20] 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.
[21] G. Chesson, “XTP PE design considerations”, *Proceedings of the IFIP Workshop on Protocols for High-Speed Networks*, Zurique, 9-11 de maio de 1989.
[22] 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.
[23] M. de Prycker, “Asynchronous Transfer Mode - Solution for Broadband ISDN”, Ed. Ellis Horwood, 1991.
[24] H. M. de Lima, “Protocolos Ponto-a-Multiponto de Alto Desempenho”, *Tese de Doutorado COPPE/UFRJ, Programa de Engenharia Elétrica*, março de 1994.