Capítulo II

Alguns protocolos de alto desempenho

Neste capítulo são analisados, em linhas gerais, os mecanismos utilizados por oito protocolos, alguns de transporte e outros que possuem esta funcionalidade: os protocolos Datakit, Delta-t, NETBLT, VMTP, XTP, ST-II, MTP e RAMP. 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. Um estudo dos protocolos Datakit, Delta-t, NETBLT, VMTP e XTP também pode ser encontrado em [Nune94b].

II.1 - Datakit

O protocolo Datakit [Ches79] [Fras89a] [Fras89b] 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 (*stream*) 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 bite 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 [ISO7498]. 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 os procedimentos 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 II.1).

Figura II.1: A estrutura do Receptor Universal.

Conexões:
Numa rede interligada pelo Datakit, as conexões são gerenciadas por estações dedicadas, utilizando *handshake* 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, a geração de reconhecimentos é submetida ao 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 bites impõe um *overhead* significativo no desempenho do sistema para implementação em *software*. Além do tratamento ser realizado *byte* a *byte*, há problemas de alinhamento decorrentes do uso de PDUs (*Protocol Data Units*) de 9 bites. A rigor, não passam de mensagens com um campo de informação de 8 bites e um cabeçalho de 1 bite. 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.

II.2 - Delta-t

O protocolo Delta-t [Wats83] [Wats89a] [Wats89b] 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.

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, discutido no capítulo anterior. Essa conclusão foi baseada no fato de que o mecanismo de *handshake* necessitaria de temporizadores nas etapas de estabelecimento e liberação da conexão para prevenir ambigüidades e a conseqüente violação das condições de segurança citadas no item I.1.1 do capítulo anterior. Outra alternativa analisada, o uso de identificadores de conexão únicos, foi descartada pois há 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. O protocolo Delta-t introduziu portanto o estabelecimento e a liberação implícitos da conexão. Entretanto, a conexão pode ser explicitamente terminada antes que os temporizadores estourem. As conexões são identificadas por uma **tripla** que consiste dos endereços da porta destino (números de 64 bites), da porta origem e de um número de conexão.

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

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*.

II.3 - NETBLT

O protocolo NETBLT (*NETwork BLock Transfer*) [Clar87a] [Clar87b] 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, os projetistas do NETBLT foram os pioneiros na utilização do controle de fluxo por taxa de transmissão. 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 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 pode haver mudanças bruscas nas condições de carga da rede, de tal forma que a informação obtida dos nós seja útil ao chegar. Isso tudo implica em modificações no processamento de nós e *gateways*, o que pode não ser viável.

II.4 - VMTP

O VMTP (*Versatile Message Transaction Protocol*) [Cher86] [Cher88] [Cher89] [Nord89] 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**.

Até o surgimento do VMTP, nenhum dos protocolos descritos neste capítulo 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.

Conexões:

O VMTP também usa o mecanismo de conexão implícita usado no Delta-t. Desta forma, uma transação de mensagens básica no VMTP (Figura II.2a) 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** (Figura II.2b), 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 multidestinatárias, sendo a conexão k-confiável com $k = 1$. 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 (Figura II.2c), 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.

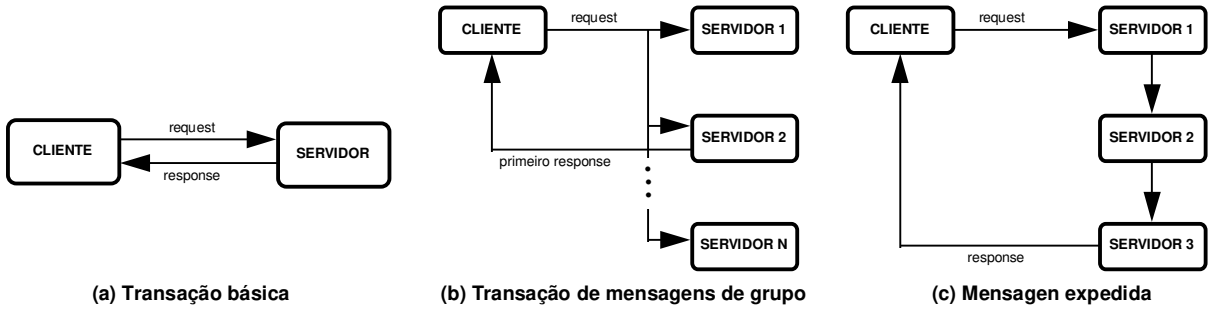

Figura II.2: Tipos de transação de mensagens no VMTP.

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 conexões ponto-a-multiponto confiáveis. 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 consiste na introdução de um atraso entre duas transmissões consecutivas. Clientes e servidores podem explicitamente comunicar os *interpacket gaps* 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 ocorrerem 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 de transbordos de memória.

II.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*) [Ches88] [Ches89] [PEI92] [Sant92] [Stra92].

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 capítulo, além dos protocolos TCP [Post81] 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 simples, visando implementação em *hardware*.

A Figura II.3 mostra a arquitetura da máquina de estados do XTP. É importante observar, entretanto, que esta é apenas uma arquitetura lógica, não representando nenhum compromisso em termos de implementação. O processo RECEPTOR atua nas mensagens recebidas 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. 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 as mensagens 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 na mensagem a ser transmitida. O processo RECEPTOR utiliza informações das mensagens recebidas para indexar a Tabela de Tradução e portanto determinar o Descritor de Contexto que guarda informações relevantes àquela mensagem.

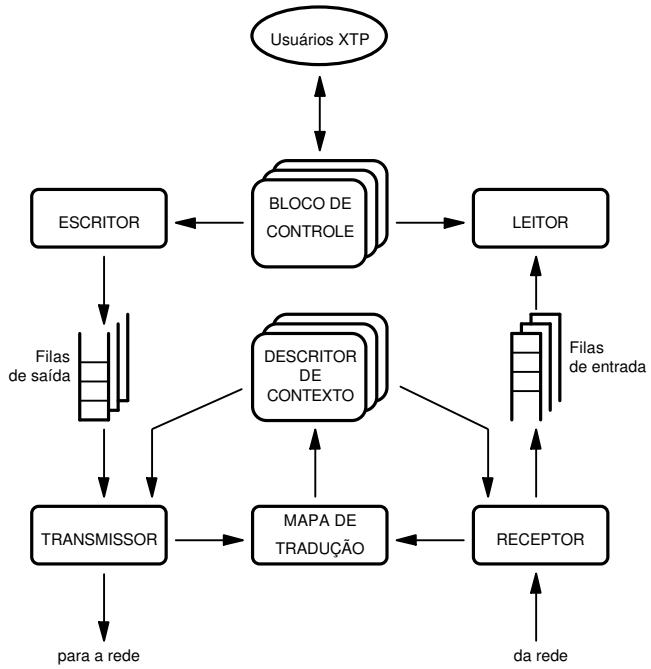

Figura II.3: Arquitetura XTP.

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 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. 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.

Conexões:

O XTP consegue confiabilidade com um estabelecimento implícito de conexões, podendo 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 octetos de dados "marcados". Estes dados são entregues ao usuário final, mas não são interpretados pelo XTP. Conexões multidestinatárias 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. 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*, que merecerão uma análise mais detalhada a seguir. 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 bites 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 bites, 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 numerados). Poderá passar facilmente para números de seqüência de 64 bites. 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, apenas o controle de fluxo por janela é fim-a-fim, considerando-se que haja um ou mais nós intermediários na conexão. O controle de fluxo por taxa é utilizado somente entre dois nós consecutivos.

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.

II.5.1 - O algoritmo *bucket*

O *bucket* é uma estrutura de dados que armazena algumas variáveis de estado relevantes à comunicação, essencialmente números de seqüência. Um temporizador denominado *switching time* (ST) é utilizado para "envelhecer" o *bucket*, em um processo que será explicado a seguir. Esse temporizador visa dar tempo suficiente para que múltiplos receptores respondam a pedidos de *status* do transmissor, sendo calculado com base em estimativas de tempo de ida e volta. Quando o temporizador ST estoura, o transmissor envia uma mensagem CNTL de pedido de reconhecimento (bite `SREQ` habilitado) e um *bucket* vazio é instanciado com o valor do campo *sync* contido no CNTL transmitido. Normalmente, *sync* recebe o valor de um contador local incrementado toda vez que uma mensagem de dados

é transmitida. Este *bucket* armazenará então informações combinadas das mensagens CNTL recebidas em resposta à solicitação do transmissor, desde que possuam o campo *echo* igual ao índice, ou *label*, do *bucket*. O transmissor deve ter um vetor de *buckets* prontos para uso. Um novo *bucket* somente é alocado após o estouro de ST, quando uma nova mensagem CNTL é transmitida e ST é reiniciado. Enquanto houver *buckets* vazios, o estouro de ST causa apenas a instanciação de um novo *bucket*. Mas quando todos os *buckets* estiverem preenchidos, a informação do *bucket* mais "velho" (com menor índice) é utilizada para atualizar o estado do transmissor, que libera parte do *buffer* de transmissão de acordo com essa informação. Este *bucket* pode assim ser reutilizado e reinicializado com o valor de *sync* contido no CNTL enviado nessa ocasião, passando a ser o *bucket* mais "novo" (com maior índice). Qualquer CNTL recebido com um valor de *echo* menor que o menor índice de *bucket* existente é imediatamente descartado. Dessa forma, o algoritmo pune os receptores com menor capacidade de processamento, conferindo uma certa homogeneidade ao grupo. Ou seja, se uma estação enviar ao transmissor um CNTL relativo a um *bucket* inexistente, pedindo a retransmissão de dados que já tenham sido liberados, ela não será atendida e seu processamento ficará bloqueado, levando-a fatalmente a deixar o grupo. Na classificação do Capítulo I, pode-se dizer que o multiponto no XTP é estatisticamente confiável.

A Figura II.4 exemplifica este procedimento, sabendo-se que:

- *rseq* é o número de seqüência imediatamente superior ao maior número de seqüência corretamente recebido;
- *dseq* é o número de seqüência imediatamente superior ao maior número de seqüência corretamente recebido e entregue ao usuário.

Na figura, o transmissor dispõe apenas de 2 *buckets*. Quando a transmissão é iniciada, ST é disparado e as mensagens de dados contendo os octetos numerados de 0 a 99 e de 100 a 199 são enviadas. Esses dados são também armazenados em *buffers*. Ao primeiro estouro de ST, uma mensagem CNTL é enviada com *sync* = 2 (pois duas mensagens de dados foram transmitidas), o primeiro *bucket* é instanciado com *label* = 2 e ST é reiniciado. As respostas começam então a chegar, na forma de duas mensagens CNTL com *echo* = 2 contendo os valores *rseq* = 200 e *dseq* = 200, que são armazenados no *bucket* 2. Uma nova mensagem DATA é transmitida, dessa vez contendo os octetos 200 a 299. Ao segundo estouro de ST, uma nova mensagem CNTL é transmitida com *sync* = 3, o segundo *bucket* é instanciado com *label* = 3 e ST é reiniciado. O transmissor recebe então um CNTL com *echo* = 2 e valores *rseq* = 100 e *dseq* = 100. Como tais valores são menores que os do *bucket* 2, eles os substituem. Após a transmissão e armazenamento dos octetos de 300 a 399, dois novos CNTL, dessa vez com *echo* = 3, são recebidos. Ambos têm o mesmo valor 300 para *rseq*, mas o segundo valor recebido de *dseq*, 300, não substituirá o anterior (200) por ser maior. Ao

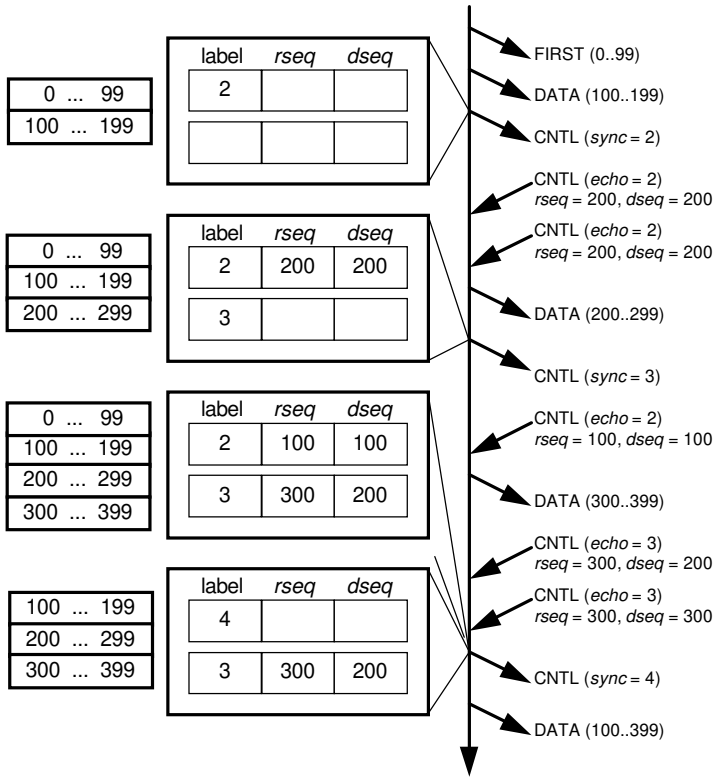

Figura II.4: Exemplo do algoritmo *bucket*.

terceiro estouro de ST, uma nova mensagem CNTL com *sync* = 4 é transmitida, mas não há *buckets* livres. Portanto, o *bucket* de menor índice deve ser reinicializado com o novo valor de *sync*, logo após sua informação ser utilizada para atualizar o estado do transmissor. Nessa ocasião, o valor *dseq* = 100 indica que, com certeza, apenas os octetos de números de seqüência até 99 foram corretamente entregues aos usuários remotos. Assim, o transmissor já pode liberar esses octetos do *buffer*. A próxima mensagem de dados deverá conter os octetos 100 a 399, configurando uma retransmissão por *go-back-N*.

II.5.2 - *Damping*

Uma vez que os receptores transmitem suas mensagens CNTL para o grupo, e não especificamente para o transmissor, todos os membros podem observar as mensagens CNTL enviadas por seus pares. Em uma comunicação livre de erros, espera-se que uma mensagem CNTL enviada por um receptor seja duplicada por todos os outros receptores em resposta a um `SREQ`. Na ocorrência de erros, receptores que detectassem erros similares enviariam mensagens CNTL similares, havendo sempre alguma porcentagem de duplicatas. O que o algoritmo *damping* faz é impedir que um receptor envie sua mensagem CNTL se alguma duplicata for detectada na rede. Esta técnica reduz a quantidade total de mensagens CNTL transmitidas pelo número de duplicatas canceladas, às custas de forçar todos os receptores a lerem todas as mensagens CNTL. Mas para grupos grandes (maiores que 10 membros), este

mecanismo pode melhorar o desempenho global, principalmente se for utilizado em conjunto com a técnica de *slotting* (ver a seguir).

II.5.3 - *Slotting*

A técnica de *slotting* impõe um atraso aleatório antes da transmissão das mensagens CNTL de reconhecimento. Entretanto, o modo de geração deste atraso é dependente da implementação. Dentre as vantagens trazidas pelo uso desta técnica, pode-se citar duas principais: em primeiro lugar, o envio de mensagens CNTL é distribuído no tempo e não mais concentrado, melhorando o desempenho no sentido em que evita uma súbita sobrecarga no tráfego da rede. Em segundo lugar, a distribuição das mensagens CNTL no tempo aumenta a possibilidade do exercício da técnica de *damping*.

II.6 - ST-II

O *Internet Stream Protocol Version II* [Topo90] é um protocolo de rede que teoricamente ocuparia o lugar do protocolo IP em ambientes Internet, com suporte a comunicações multidestinatárias. Mas apesar de ser um protocolo de rede, incorpora algumas funcionalidades de transporte, pois foi projetado para garantir banda-passante e um valor máximo para o atraso fim-a-fim.

Conexões:
O estabelecimento de uma conexão no ST-II envolve a negociação de banda passante, parâmetros de atraso (mínimo, máximo e variação) e da rota entre o transmissor, os nós intermediários e os receptores. Se os recursos solicitados não estiverem disponíveis, é dada ao iniciador a opção de aceitar o que puder ser oferecido ou cancelar o pedido de conexão. Uma nova conexão somente é estabelecida se houver recursos suficientes para garantir as exigências de banda-passante e atraso. Para adicionar um usuário a um grupo já em andamento, o receptor deve se comunicar diretamente com o iniciador da conexão para obter o endereço do grupo, entre outras informações relevantes. A aplicação no transmissor pede então à camada ST-II subjacente para adicionar o novo receptor ao grupo. Conexões no ST-II podem ser fechadas pelo transmissor ou pelo receptor. Quando o último receptor ao longo de um caminho é removido, os recursos alocados para aquele caminho são liberados. Quando todos os receptores do grupo forem removidos, o transmissor é informado e tem a opção de adicionar um novo receptor ou terminar o grupo.

Reconhecimentos:
A mensagem ACK no ST-II é utilizada para reconhecer algumas mensagens de controle, como de aceitação de conexão (mensagem ACCEPT), alteração de um ou mais parâmetros do *FlowSpec* (mensagem CHANGE), pedido de estabelecimento de conexão (mensagem CONNECT), pedido de desconexão (mensagem DISCONNECT), pedidos de *status* dos receptores (mensagem HELLO), informação da origem ou destino de algum evento significativo (mensagem NOTIFY) e rejeição de conexão (mensagem REFUSE).

Controle de fluxo:
O controle de fluxo no ST-II é por taxa de transmissão, determinada a partir de alguns parâmetros do *FlowSpec*. O *FlowSpec* é um conjunto de parâmetros de qualidade de serviço utilizado na alocação de recursos para uma conexão.

Controle de erros:
O ST-II utiliza retransmissão por temporização.

II.7 - MTP

O *Multicast Transport Protocol* [Arms92] é um protocolo de transporte especialmente projetado para oferecer conexões ponto-a-multiponto confiáveis se aproveitando dos serviços de endereçamento de grupo do protocolo IP Multiponto [Deer89]. Ele se baseia no conceito de um centralizador, ou **mestre**, que controla todos os aspectos da comunicação. Existem três classes de entidades no MTP: os **consumidores**, que podem apenas receber dados, os **produtores**, que podem transmitir e receber, e o mestre, um superconjunto do produtor.

Como pode haver vários produtores em um grupo, o MTP adota um mecanismo de **fichas** para que todos tenham a oportunidade de transmitir dados. O produtor interessado em enviar ao grupo uma mensagem deve pedir a ficha (existe apenas uma) diretamente ao mestre. Se a ficha estiver em poder de algum outro produtor, o pedido é colocado em uma fila FIFO (esquema de prioridade mais simples sugerido para implementação). Quando chegar sua vez de ser atendido, o produtor obtém a ficha, que deve ser devolvida ao final da transmissão da mensagem ou ao estouro de um temporizador fixado pelo mestre.

Conexões:
Uma vez que o endereço de um grupo é estabelecido (o MTP não tem nenhum controle sobre este evento), qualquer estação pode enviar ao mestre um pedido de adesão ao grupo, especificando se deseja ser produtor ou consumidor e parâmetros de qualidade de serviço, como vazão mínima e tamanho máximo das mensagens. Se as exigências puderem ser

atendidas, o mestre envia ao novo membro uma resposta contendo o endereço do grupo. Caso um participante deseje deixar o grupo, este também deve enviar uma notificação ao mestre.

Reconhecimentos:
O MTP utiliza reconhecimentos negativos para informar as lacunas (*gaps*) ocorridas na recpeção.

Controle de fluxo:
O MTP utiliza um mecanismo de controle de fluxo por taxa baseado em rajadas e temporizadores, ou *windows* e *heartbeats* em sua nomenclatura.

Controle de erros:
A retransmissão no MTP é seletiva, sendo que as mensagens são retransmitidas a todos os membros do grupo.

II.8 - RAMP

O *Reliable Adaptive Multicast Protocol* [Brau93] é um protocolo de transporte projetado também para oferecer conexões ponto-a-multiponto confiáveis. O projeto parte da premissa de que transmissões confiáveis ocorrem às custas de um certo grau de perda de qualidade de serviço (QOS). O protocolo RAMP permite que um serviço seja oferecido em diferentes níveis de QOS. Um campo no cabeçalho das mensagens indica os níveis de QOS associados àquela mensagem. O QOS solicitado pode ser modificado durante a transferência de dados.

Conexões:
Existe internamente ao RAMP um protocolo unicamente responsável pelo gerenciamento de endereços, o registro de serviços e a manutenção de grupos, o MGA (*Multicast Group Authority*). O MGA mantém informações de estado sobre todos os serviços multidestinatários registrados e todos os receptores, incluindo o número de membros em cada grupo, que é formado de acordo com parâmetros de QOS compatíveis. Os servidores registram a disponibilidade de um serviço, especificando o grau de confiabilidade e o nível de QOS oferecidos. Um endereço de grupo é alocado e um identificador atribuído. Um cliente solicita um serviço especificando os parâmetros desejados de QOS. Se o serviço estiver registrado e atender a tais parâmetros, uma conexão é estabelecida para servir ao novo usuário. Caso contrário, o pedido é colocado em uma fila até que aquele serviço seja registrado. Pedidos de término de um grupo são também processados pelo MGA.

Reconhecimentos:
O RAMP utiliza reconhecimentos negativos para informar os números de seqüência de mensagens não recebidas ou incorretas.

Controle de fluxo:
O RAMP realiza um controle de fluxo por taxa de uma maneira dinâmica que reduz a taxa baseando-se em pedidos de retransmissão ou em pedidos explícitos de diminuição de taxa feitos pelo roteador, e aumenta a taxa indiretamente através do número de mensagens transmitidas corretamente, ou seja, que não tenham sido retransmitidas.

Controle de erros:
O RAMP utiliza números de seqüência para ordenação de mensagens, mas também inclui a opção de entrega de mensagens fora de ordem. O mecanismo de retransmissão adotado é o seletivo. A entidade RAMP transmissora acumula pedidos de retransmissão durante um determinado intervalo de tempo. Ao final deste intervalo, é feita uma comparação do número de pedidos de retransmissão de uma determinada mensagem com um valor limite, ou *threshold*. Todas as mensagens cujos pedidos de retransmissão ultrapassarem este valor são retransmitidas para todo o grupo, enquanto as demais são transmitidas ponto-a-ponto para cada receptor. O temporizador de retransmissão é aumentado ou diminuído de acordo com o aumento ou diminuição, respectivamente, do número de pedidos de retransmissão.