MCF: Um Protocolo *M*ultidestinatário *C*onfigurável com Gerenciamento de Acesso por *F*ichas de Permissão[1]

Marcelo Dias Nunes[2]
email:bill@coe.ufrj.br

Otto Carlos M. B. Duarte
email: otto@coe.ufrj.br

Grupo de Teleinformática e Automação - GTA
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

Este artigo apresenta a proposta e a implementação de um protocolo configurável com suporte à comunicações multidestinatárias estatisticamente confiáveis e com gerenciamento centralizado de acesso através da concessão de fichas de permissão, apropriado para ambientes de redes locais. Os serviços oferecidos incluem abertura e fechamento de conexões ponto-a-ponto, bem como adesão a grupos multiponto, com seleção de mecanismos (cálculo de *checksum*, controle de fluxo por janela e taxa de transmissão, recuperação de erros por *go-back-N* e retransmissão seletiva, algoritmo *bucket*), utilizando os serviços da sub-camada MAC nas configurações ponto-a-ponto e ponto-a-multiponto no ambiente de desenvolvimento. A implementação segue uma arquitetura de alto desempenho e os as medidas de vazão efetuadas comprovam uma alta eficiência além de uma grande flexibilidade na escolha de opções dos mecanismos fornecidos.

Abstract

This paper presents a proposal and an implementation of a programmable protocol that supports statistically reliable multipoint communications, with a token-based centralized access management well suited for LAN environments. Among other services, it offers pointo-to-point connection establishment and release, as well as multicast group membership, with mechanism selection (checksum computing, window and rate-based flow control, error recovery through go-back-N and selective retransmission, bucket algorithm), using services provided by the MAC sublayer in point-to-point and multicast configurations on the development environment. The implementation follows a high performance architecture. The results show that the protocol attains a high throughput and fits well for different traffic types.

---

[1]Este trabalho foi realizado com recursos da UFRJ, FUJB, CAPES, CNPq PROTEM-CC.
[2]Aluno de doutorado do Programa de Engenharia Elétrica da COPPE/UFRJ.

1 - Introdução

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 centenas de Mbps. Espera-se que em um futuro próximo as redes de longa distância (WANs) alcancem velocidades de gigabites por segundo. Decididamente, o fator limitante tranferiu-se da banda-passante para o processamento do protocolo, o que impede uma aplicação de utilizar uma fração razoável da banda disponível.

Além disso, as Redes Digitais de Serviços Integrados em Banda Larga (RDSI-BL) deverão oferecer serviços que suportem uma vasta gama de aplicações: transferência de arquivos, transações cliente/servidor, datagrama, transferências multidestinatárias e aplicações de tempo real.

Nos últimos anos, vários protocolos têm sido propostos para atender aos requisitos de alta velocidade, sendo conhecidos como *light-weight protocols* [Doer90, Nune94b e Nune95a]. Entre eles, destacam-se os seguintes protocolos e suas principais contribuições:

- Datakit [Ches79]: introduziu o conceito de envio de reconhecimentos submetido a comandos do transmissor;
- Delta-t [Wats89]: introduziu o conceito de estabelecimento e liberação implícitos de conexão;
- NETBLT [Clar87]: introduziu o mecanismo de controle de fluxo por taxa de transmissão, através da determinação da rajada (em octetos) e da taxa (em octetos por segundo);
- VMTP [Cher89]: protocolo orientado a transações (arquitetura cliente/servidor);
- XTP [PEI92]: protocolo de processamento simplificado visando implementação em *hardware*; utiliza o algoritmo *bucket* para controle de erros em conexões ponto-a-multiponto. Este protocolo tem o grande mérito de adotar o que se tinha de melhor nos protocolos precedentes;
- ST-II [Topo90] e RSVP [Zhan93]: permitem negociação de parâmetros de qualidade de serviço utilizando mecanismos de reserva de recursos;
- MTP [Arms92]: protocolo ponto-a-multiponto com gerenciamento de acesso por ficha;
- RAMP [Brau93]: permite que um serviço seja oferecido em diferentes níveis de qualidade de serviço de acordo com o nível de confiabilidade desejado.

Todos são protocolos que, de uma forma ou de outra, procuram simplificar o processamento para atingir alto desempenho. Porém, cada um tem suas características específicas, tornando-os mais adequados a uma ou outra aplicação particular.

Portanto, o presente artigo define um serviço e especifica um protocolo multidestinatário estatisticamente confiável, configurável e com gerenciamento de acesso por fichas de permissão. Entretanto, uma das etapas fundamentais na definição deste novo protocolo foi o estudo dos mecanismos de comunicação usados nos demais protocolos de alto desempenho supra-citados:

- **conexões**, incluindo 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 transbordo de memória.

O protocolo proposto foi desenvolvido seguindo uma arquitetura de implementação de alto desempenho [Nune95b], tendo sido alcançados resultados de vazão significativos.

A seção 2 apresenta as decisões de projeto, tomadas com base em análise prévia dos diferentes mecanismos de gerenciamento de conexões, reconhecimentos, controle de fluxo e controle de erros. A seção 3 apresenta as principais características do protocolo proposto: primitivas de serviço, formato de PDUs e as principais tabelas de estado. A seção 4 ilustra o sistema implementado e a seção 5 descreve de maneira sucinta a metodologia de medidas para análise de desempenho e os principais resultados obtidos. Por fim, conclusões são apresentadas na seção 6.

2 - Decisões de projeto

As facilidades visadas incluem transferência de informações de uma forma confiável, estatísticamente confiável e não confiável em configurações ponto-a-ponto e multidestinatárias. Além disto, objetiva-se oferecer uma série de opções de mecanismos para atender uma grande gama de aplicações que geram diferentes tipos de tráfego.

Para se chegar a uma conclusão acerca dos mecanismos mais adequados a um serviço flexível, com conexões ponto-a-ponto confiáveis e multidestinatárias estatisticamente confiáveis, foi feita uma análise comparativa de diferentes mecanismos de gerenciameento de conexões, reconhecimentos, controle de fluxo e controle de erros. Esta seção apresenta as decisões de projeto tomadas com base nesta análise. Por questões de simplicidade, o protocolo proposto neste artigo será doravante referenciado pela sigla que resume suas principais características: **MCF** (protocolo **M**ultidestinatário **C**onfigurável com gerenciamento de acesso por **F**ichas de permissão).

- **Gerenciamento de conexões**: para o protocolo MCF optou-se por adotar o mecanismo de *handshake* para abertura e fechamento de conexões ponto-a-ponto. Conexões multidestinatárias utilizam uma espécie de estabelecimento "implícito", porém sem temporizadores para manutenção da conexão. A principal vantagem do *handshake* que pesou nessa decisão é a possibilidade de negociar **parâmetros** da conexão, como tipo de controle de fluxo e de erros, pois a diversidade das aplicações atuais com diferentes exigências de banda-passante e confiabilidade, entre outras, pede um protocolo **configurável**, a exemplo do protocolo XTP.

- **Reconhecimentos**: devido à complexidade introduzida pela manipulação de temporizadores, obrigatórios quando se utiliza geração de reconhecimentos independente do transmissor, os quais seriam mais necessários em outros mecanismos, como será visto adiante, optou-se pela geração de reconhecimentos submetida à comandos explícitos do transmissor. Esta técnica simplifica o processamento no receptor, naturalmente prejudicado por exercer um maior processamento por mensagem que o transmissor.
- **Controle de Fluxo**: a utilização de janelas de transmissão, de tamanho fixo ou não, para controle de fluxo é, acima de tudo, um mecanismo de controle da utilização dos *buffers* de transmissão e, principalmente, de recepção, para evitar problemas de transbordo de memória. Por sua vez, o controle de fluxo através do controle da taxa de transmissão é aparentemente melhor habilitado a suportar diferentes padrões de geração de tráfego, contanto que a granularidade da temporização permita um mínimo de precisão na taxa efetivamente obtida, e que a rede subjacente possa interagir com o protocolo para a atualização de valores de taxa e rajada. Caso contrário, tais parâmetros devem ser negociados na fase de estabelecimento da conexão. Em suma: ambos os mecanismos são válidos, pois endereçam diferentes aspectos da comunicação e são portanto oferecidos na implementação do protocolo MCF.
- **Controle de Erros**: foi decidido deixar para a aplicação a opção do modo de recuperação de erros que a ela melhor se adapte. Logo, o protocolo MCF suporta os dois mecanismos convencionais de recuperação de erros, *go-back-N* e retransmissão seletiva, com disparo de retransmissões através de pedido de repetição automática (ARQ). Além disso, números de seqüência nas PDUs são utilizados para detecção de perdas e *checksums* são utilizados para detecção de erros, separadamente para o cabeçalho e para o segmento de dados (este último *checksum* é opcional na PDU de dados do usuário).
- **Semântica das Conexões Multidestinatárias**: o protocolo MCF implementa conexões estatisticamente confiáveis. Assim, pelo menos teoricamente, um grupo tem tamanho ilimitado, pois nenhum membro, seja ele transmissor ou receptor, precisa conhecer o número total de membros ou seu *status*. Basta conhecer o endereço do grupo. O algoritmo *bucket* [Sant92] permite a implementação de conexões estatisticamente confiáveis, tendo sido portanto utilizado no protocolo MCF. Para este protocolo foi criado também um mecanismo de fichas que permite a um subconjunto do grupo atuar simultaneamente como transmissor. São utilizadas *N* fichas, onde *N* é menor do que o número máximo de participantes da comunicação (limitado, na prática, pela memória disponível).

3 - Definição do serviço e especificação do protocolo MCF

3.1 - Primitivas de serviço

As facilidades oferecidas pelo serviço proposto são definidas em forma de primitivas de serviço. O serviço ponto-a-ponto pode ser dividido em três fases:

abertura e fechamento de conexões e transferência de dados. O serviço multiponto não prevê uma conexão com troca de mensagens para não inviabilizar um grande número

Tabela 1: Primitivas de serviço.

Primitivas	Descrição	Parâmetros
T-CONNECT.Request	Pedido de estabelecimento de conexão ponto-a-ponto	Endereço Destino Endereço Origem Qualidade de Serviço
T-CONNECT.Indication	Indicação de pedido de estabelecimento de conexão ponto-a-ponto	Endereço Destino Endereço Origem Qualidade de Serviço
T-CONNECT.Response	Resposta ao pedido de estabelecimento de conexão ponto-a-ponto	Endereço Respondedor Resposta (aceitação ou recusa) Qualidade de serviço
T-CONNECT.Confirm	Confirmação do pedido de estabelecimento de conexão ponto-a-ponto	Endereço Respondedor Resposta Qualidade de Serviço
T-DISCONNECT.Request	Pedido de liberação de conexão ponto-a-ponto	Tipo de desconexão (normal ou aborto)
T-DISCONNECT.Indication	Indicação de pedido de liberação de conexão ponto-a-ponto	Razão de desconexão
T-DISCONNECT.Response	Resposta ao pedido de liberação de conexão ponto-a-ponto	não tem
T-DISCONNECT.Confirm	Confirmação do pedido de liberação de conexão ponto-a-ponto	não tem
T-DATA.Request	Pedido de transmissão de dados do usuário	Dados do usuário
T-DATA.Indication	Indicação de recepção de dados do usuário	Dados do usuário
T-MULTIPOINT.Request	Adesão a um grupo multiponto	Categoria (servidor ou não) Qualidade de serviço
T-TOKEN.Request	Pedido de ficha para transmissão em grupos multiponto	Endereço Destino Endereço Origem
T-TOKEN.Indication	Indicação de recepção ou devolução de ficha	não tem
T-TOKEN.Response	Comando de devolução da ficha	não tem
T-LEAVE.Request	Desligamento de um grupo multiponto	não tem

de entidades comunicantes. Este serviço presupõe que a camada inferior difunda a informação através de um endereçamento de grupo (na implementação foi usado o protocolo *Medium Access Control* -MAC com endereçamento de grupo). Duas primitivas são usadas para indicar a adesão e o desligamento ao serviço.

Uma facilidade bastante original oferecida pelo serviço é o gerenciamento de permissões de transmissão. Com a finalidade de permitir uma conferência com um número bastante amplo de usuários o serviço provê um gerenciamento centralizado de múltiplas fichas. Isto tem a finalidade de ordenar a conferência não permitindo que "todos falem ao mesmo tempo". Por conter algumas funcionalidades características da Camada de Transporte do Modelo OSI/ISO (como segmentação e remontagem, juntamente com garantias de confiabilidade fim-a-fim) as primitivas seguem o Modelo OSI para o serviço de Transporte. A Tabela 1 mostra todas as primitivas de serviço definidas com uma breve descrição e os parâmetros de cada uma delas.

3.2 - O Protocolo MCF

Muitas das características exigidas por um protocolo de alto desempenho foram adotadas pelo protocolo XTP. Estas características incluem o cabeçalho de tamanho fixo, verificação de erros em separado para cabeçalho e campo de dados, possibilidade de conexão implícita sem troca de mensagens, mecanismo de *bucket,* alinhamento em 32 bites, etc. Assim, resolveu-se adotar, sempre que possível a formatação e denominação dos tipos de PDUs (*Protocol Data Units*) deste protocolo acrescentando e adaptando quando necessário.

3.2.1 Formato das PDUs

3.2.1.1 - Cabeçalho

Todas as PDUs utilizadas por este protocolo contém o mesmo cabeçalho, de tamanho fixo, com sete campos de 32 bites, mostrado na Figura 1.

Campo** cmd* **- *Command: o campo *cmd* (Figura 2) contém informações quanto ao tipo da PDU (1 octeto), o número de octetos (1 octeto) utilizados para completar o campo de dados do usuário (em PDUs FIRST ou DATA), tornando seu comprimento múltiplo de 32 bites (*padding*), e as opções de configuração do protocolo (2 octetos).

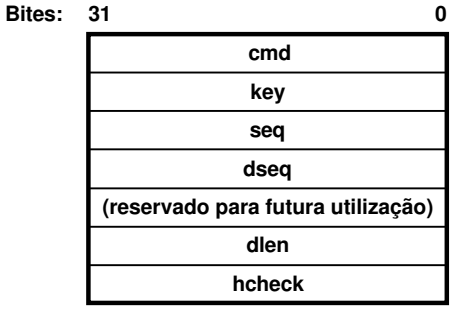

Figura 1: Cabeçalho das PDUs.

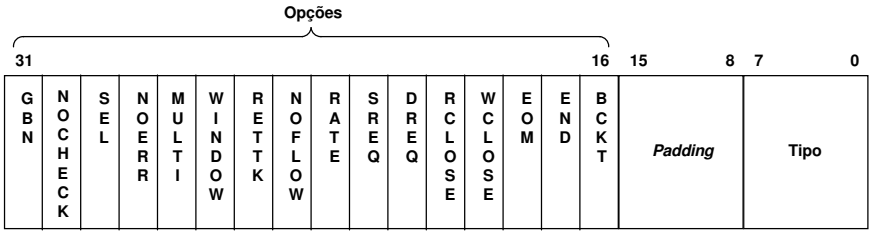

Figura 2: Campo *cmd*.

Tipos de PDU:

- 0x00: DATA PDU de dados do usuário;
- 0x01: CNTL PDU de informações de controle, servindo como PDU de reconhecimento;
- 0x02: FIRST PDU de estabelecimento de conexão ponto-a-ponto (também pode conter dados do usuário);
- 0x04: TOKEN PDU representativa da ficha em conexões multidestinatárias;
- 0x06: PASS PDU de pedido de ficha ao servidor;
- 0x08: ERROR PDU de diagnóstico em situações de erro.

Opções de configuração:

- `GBN`**:** Habilitado, indica opção pelo mecanismo *go-back-N* para recuperação de mensagens.
- `NOCHECK` (*Suspend CRC checking*): Habilitado, indica que o *checksum* sobre os **dados** não deve ser calculado.
- `SEL` (*Selective retransmission*): Habilitado, indica opção pelo mecanismo de retransmissão seletiva para recuperação de mensagens.
- `NOERR` (*Suspend retransmission*): Habilitado, indica que nenhum controle de erros será realizado. Nesse caso, os bites `GBN`, `SEL` e `BCKT` (para conexões multidestinatárias) não são considerados.
- `MULTI` (*Multicast operation*): Habilitado, indica conexão multidestinatária.
- `WINDOW`**:** Habilitado, indica opção pelo mecanismo de janelas de transmissão para controle de fluxo.
- `RETTK` (*Return Token*): Habilitado em PDUs CNTL enviadas pelo servidor para que uma determinada estação devolva a ficha que está em seu poder.
- `NOFLOW` (*Suspend flow control*): Habilitado, indica que nenhum controle de fluxo será realizado. Nesse caso, os bites `WINDOW` e `RATE` não são considerados.
- `RATE`: Habilitado, indica opção pelo mecanismo de controle de fluxo através da taxa de transmissão.
- `SREQ` (*Status request*): Habilitado em PDUs DATA ou CNTL, indica que a entidade destinatária deve enviar imediatamente uma PDU CNTL como resposta. Serve portanto como pedido de reconhecimento pela entidade transmissora.
- `DREQ` (*Delivery status request*): Semelhante ao `SREQ`, com a diferença que a PDU CNTL é somente enviada após a entrega, ao usuário, de quaisquer mensagens corretamente recebidas que ainda estejam armazenadas em *buffers*.
- `RCLOSE` (*Reader closed*): Utilizado no fechamento de uma conexão ponto-a-ponto, indica à entidade remota o "fechamento do canal de leitura", ou seja, PDUs DATA não serão mais aceitas.

- `WCLOSE` (*Writer closing*): Análogo ao anterior, indica o "fechamento do canal de escrita", ou seja, PDUs DATA não serão mais transmitidas.
- `EOM` (*End of message*): Deve ser habilitado na última PDU DATA de uma mensagem.
- `END`**:** Habilitado, indica o fechamento definitivo da conexão.
- `BCKT`: Habilitado, indica a opção pelo mecanismo de *buckets* para controle de erros em conexões multidestinatárias.

***Campo** key*: representa o número da conexão.

Campo** seq* **- *Current sequence number: o campo *seq* representa o número de seqüência de uma PDU DATA. Este campo deve conter o valor **zero** em qualquer outra PDU.

Campo** dseq* **- *Delivered sequence number: o campo *dseq* também deve ser interpretado em PDUs DATA e CNTL, e representa o número de seqüência imediatamente superior ao da última PDU DATA cujos dados foram realmente entregues ao usuário. Quando recebido pelo transmissor, indica que todas as PDUs armazenadas em *buffers* para retransmissão, com números de seqüência até *dseq* **exclusive**, podem ser liberadas.

Campo** dlen* **- *Data length: o campo *dlen* representa o número total de octetos do segmento de dados. No caso de segmentos de informação (PDUs FIRST e DATA), inclui também quaisquer octetos de *padding* que tenham sido acrescentados para tornar o campo de dados do usuário múltiplo de 32 bites.

Campo** hcheck* **- *Header checksum: o campo *hcheck* contém o cálculo da função de *checksum* sobre os sete campos de 4 octetos do cabeçalho. Esse *checksum* é obrigatório e não pode ser desabilitado pelo bite `NOCHECK`.

3.2.1.2 - Cauda

A cauda das PDUs consiste em apenas um campo de 4 octetos, *dcheck*. O campo *dcheck* corresponde à função de *checksum* aplicada aos octetos do segmento de dados da PDU, de tamanho variável, com diferentes formatos para diferentes PDUs. Por exemplo, em PDUs DATA o segmento de dados consiste apenas dos dados do usuário, enquanto que em PDUs CNTL o segmento de dados contém vários campos correspondentes às variáveis de estado relevantes à conexão, como números de seqüência, parâmetros de taxa de transmissão, etc. Nas PDUs FIRST, CNTL, PASS, TOKEN e ERROR, o cálculo de *dcheck* é obrigatório. Apenas na PDU DATA o seu cálculo é facultativo, sendo regido pela configuração do bite `NOCHECK`. A Figura 3 mostra o formato geral de uma PDU.

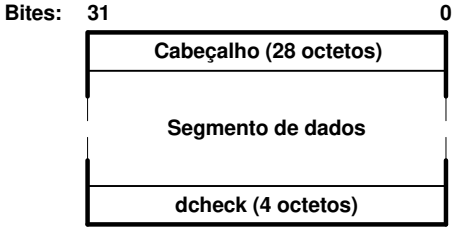

Figura 3: Formato de uma PDU.

3.2.1.3 - Segmento de dados de PDUs FIRST

Campo *addr_control* **- *Address Control***: o campo *addr_control* subdivide-se em dois outros campos: *service* e *addr_format*. O campo *service* é usado para especificar o modo de operação da comunicação. Na versão atual, apenas dois modos estão disponíveis: modo 0, para conexões ponto-a-ponto, e modo 1, para conexões multidestinatárias. O campo *addr_format* identifica a sintaxe de endereçamento.

Campo *rate*: o campo *rate* das PDUs FIRST permite ao iniciador da conexão sugerir à entidade remota um valor inicial para a taxa de transmissão (em octetos por segundo), que poderá ser aceita ou não.

Campo *burst*: o campo *burst* das PDUs FIRST permite ao iniciador da conexão sugerir à entidade remota um valor inicial para a rajada de mensagens (em octetos), que poderá ser aceita ou não.

Campo *dst_addr* **- *Destination address***: o campo *dst_addr* contém o identificador lógico da entidade respondedora da conexão.

Campo *src_addr* **- *Source address***: o campo *src_addr* contém o identificador lógico da entidade iniciadora da conexão.

Campo de dados: o campo de dados tem tamanho variável e é utilizado para transmitir os dados do usuário. Esse campo é opcional em PDUs FIRST.

3.2.1.4 - Segmento de dados de PDUs DATA

É na PDU DATA onde os dados do usuário são efetivamente transmitidos. Seu segmento de dados consiste apenas no campo de dados descrito no item anterior.

3.2.1.5 - Segmento de dados de PDUs CNTL

Campo *rate*: o campo *rate* das PDUs CNTL permite ao respondedor da conexão sugerir à entidade iniciadora uma contra-proposta para a taxa de transmissão (em octetos por segundo), que poderá ser aceita ou não.

Campo *burst*: o campo *burst* das PDUs CNTL permite ao respondedor da conexão sugerir à entidade iniciadora uma contra-proposta para a rajada de mensagens (em octetos), que poderá ser aceita ou não.

Campo *sync* **- *Control synchronization***: o campo *sync* é usado para permitir a sincronização de estado entre duas ou mais entidades através da troca de PDUs CNTL.

Campo *echo* **- *Control echo***: o campo *echo* é utilizado em conjunto com *sync* para permitir a sincronização de estado entre duas ou mais entidades. Quando o receptor envia uma PDU CNTL em resposta à alguma solicitação do transmissor, o valor de *sync* recebido deve ser copiado para o campo *echo*. Dessa forma, o transmissor poderá associar aquela resposta ao seu pedido de *status* ou reconhecimento.

Campo** time **- Round trip time: o campo *time* é usado para calcular uma estimativa do tempo de ida-e-volta (*round trip time*) através de uma operação de sincronização por meio do par *sync*/*echo*.

Campo** techo **- Time echo: o campo *techo* é usado em conjunto com *time* para calcular uma estimativa do tempo de ida-e-volta (*round trip time*) através de uma operação de sincronização por meio do par *sync*/*echo*. Quando o receptor envia uma PDU CNTL em resposta a algum pedido do transmissor, o valor de *time* recebido é copiado para o campo *techo*. No transmissor, a diferença entre *techo* recebido e *time* enviado corresponde à estimativa do tempo de ida-e-volta.

Campo** rseq **- Received sequence number: o campo *rseq*, como já foi dito anteriormente, representa o número de seqüência imediatamente superior ao número de seqüência da última PDU DATA corretamente recebida. Diferentemente de *dseq*, entretanto, não significa que os dados até este ponto tenham sido entregues ao usuário.

Campo** num_msgs **- Number of message sequence numbers: o campo *num_msgs* contém o total de números de seqüência de PDUs DATA presente no campo *msgs*.

Campo** msgs **- Message sequence numbers: o campo *msgs* é utilizado pelas entidades receptoras para a recuperação de erros através de retransmissões seletivas. De tamanho variável, consiste em uma lista dos números de seqüência das PDUs DATA que tenham sido incorretamente recebidas ou perdidas. A quantidade desses números deve constar do campo *num_msgs*.

3.2.1.6 - Segmento de dados de PDUs TOKEN

Seu segmento de dados consiste simplesmente nos campos *dst_addr* e *src_addr*, discutidos no item III.1.3.

3.2.1.7 - Segmento de dados de PDUs PASS

Seu segmento de dados é igual ao da PDU TOKEN.

3.2.1.8 - Segmento de dados de PDUs ERROR

Campo** e-code **- Error code: esse campo especifica as condições de erro que causam o envio de PDUs ERROR (operação inválida, destino desconhecido, erro de protocolo, conexão abortada, etc.).

Campo** e-val **- Error value: o campo *e-val* destina-se a fornecer informações complementares para cada código definido em *e-code*. Não está implementado nesta versão.

3.2.2 - Máquina de Estados Finita para abertura de conexões ponto-a-ponto

Estados:

- Idle: estado de repouso, conexão fechada.
- Listen: lado respondedor da conexão espera resposta do usuário.
- Connect-Conf: lado iniciador da conexão espera resposta da entidade par.
- Active: estado de transferência de dados, conexão aberta.

Predicados:

- p01: aceitação do pedido de conexão.

Ações:

- [01]: configurar opções da conexão.
- [02]: enviar T-CONNECT.Indication.
- [03]: enviar T-CONNECT.Confirm.
- [04]: enviar T-DISCONNECT.Indication.
- [05]: enviar FIRST.
- [06]: enviar CNTL.
- [07]: enviar ERROR(*e-code* = 1)

Tabela 2: Estabelecimento de conexão ponto-a-ponto

ESTADOS EVENTOS	IDLE	LISTEN	CONNECT-CONF
T-CONN.Req	[01] [05] CONNECT-CONF		
T-CONN.Rsp		p01 [01] [06] ACTIVE ^p01 [07] IDLE	
FIRST	[01] [02] LISTEN		
CNTL			[01] [03] ACTIVE
ERROR(*e-code*=1)			[04] IDLE

3.2.3- Abertura de conexões multiponto

Não existe propriamente um estabelecimento de conexões multidestinatárias. Um dos principais objetivos deste protocolo é oferecer um serviço de comunicações de grupo **dinâmico**, ou seja, a qualquer momento estações podem entrar ou sair de um grupo, sem que isso afete a comunicação. O protocolo MCF utiliza um gerenciamento

de grupo centralizado. Cada grupo deve ter um servidor, ou centralizador, de fichas de transmissão, cujo endereço lógico é bem conhecido por todas as estações da rede local.

No protocolo MCF, o servidor de fichas define um endereço de grupo de acordo com a rede subjacente (IP Multiponto, Ethernet), que também deve ser de conhecimento geral. Portanto, qualquer estação de posse desse endereço pode transmitir dados para o grupo (uma vez que a ficha lhe tenha sido concedida) e receber dados direcionados ao grupo, desde que esteja configurada para tanto, sem que nenhuma conexão seja estabelecida.

3.2.4 - Máquina de Estados Finita para fechamento de conexões ponto-a-ponto

Estados:

- Close: lado iniciador da desconexão espera resposta da entidade remota.
- Close-Conf: lado respondedor da desconexão espera resposta do usuário.

Predicados:

- p02: desconexão normal.
- p03: aborto.

Ações:

- [08]: enviar T-DISCONNECT.Confirm.
- [09]: enviar CNTL(`WCLOSE`, `RCLOSE`, `DREQ`).
- [10]: enviar CNTL(`WCLOSE`, `RCLOSE`, `END`).
- [11]: enviar CNTL(`END`).

Tabela 3: Liberação de conexão ponto-a-ponto.

ESTADOS EVENTOS	ACTIVE	CLOSE-CONF	CLOSE
T-DISCONN.Req	p02 [09] CLOSE p03 [11] IDLE		
T-DISCONN.Rsp		[10] IDLE	
CNTL(`WCLOSE`, `RCLOSE`, `DREQ`)	[04] CLOSE-CONF		
CNTL(`END`)	[04] IDLE		
CNTL(`WCLOSE`, `RCLOSE`, `END`)			[08] IDLE

3.2.5 - Fechamento de conexões multiponto

Da mesma forma que não há uma etapa explícita de estabelecimento de conexões multidestinatárias, não há também uma etapa de liberação. Um usuário pode simplesmente deixar um grupo, através da primitiva T-LEAVE.Request, o que coloca a máquina de estados diretamente em IDLE. Nenhuma notificação de abandono é enviada ao servidor ou a qualquer membro do grupo.

3.2.6 - Máquinas de Estado Finitas para gerenciamento da transmissão em um grupo através da concessão de fichas

Estados:

- Active: em conexões multidestinatárias, uma entidade estará em Active com ou sem a ficha de transmissão. Ela poderá sempre receber dados, mas somente poderá transmiti-los se a variável *Token_Granted* for verdadeira.
- Wait-Token: estado de espera pela ficha; dados podem ser recebidos.

Predicados:

- p04: *Token_Granted* = FALSE.
- p05: endereço não consta da fila de pedidos.
- p06: primeiro endereço da fila de pedidos = endereço local.
- p07: fila de pedidos não cheia.
- p08: fila de pedidos não vazia.
- p09: número de fichas no servidor > 0.
- p10: número de fichas no servidor < máximo.
- p11: contador de tentativas > 0.
- p12: endereço associado a TTIMER = endereço local.

Ações:

- [12]: atribuir FALSE a *Token_Granted*.
- [13]: atribuir TRUE a *Token_Granted*.
- [14]: colocar endereço na fila de pedidos.
- [15]: disparar temporizador TTIMER.
- [16]: interromper temporizador TTIMER.
- [17]: disparar temporizador WTIMER.
- [18]: atribuir 3 ao contador de tentativas.
- [19]: decrementar contador de tentativas.
- [20]: interromper temporizador WTIMER.
- [21]: interromper temporizador associado à ficha recebida (pode ser TTIMER ou WTIMER).
- [22]: incrementar número de fichas no servidor.
- [23]: decrementar número de fichas no servidor.
- [24]: enviar T-TOKEN.Indication.
- [25]: enviar CNTL(`RETTK`).
- [26]: enviar PASS.
- [27]: enviar TOKEN.
- [28]: enviar ERROR(*e-code* = 7).
- [29]: enviar ERROR(*e-code* = 9).
- [30]: enviar ERROR(*e-code* = 10).

Tabela 4: Gerenciamento de fichas no produtor.

ESTADOS EVENTOS	ACTIVE	WAIT-TOKEN
T-TOKEN.Req	[26] WAIT-TOKEN	
T_TOKEN.Rsp	[27] [12]	
CNTL(`RETTK`)	p04 [30] [24] ^p04 [27] [24] [12]	[29] [24] ACTIVE
TOKEN		[24] [13] ACTIVE
ERROR(*e-code* = 7)		[24] ACTIVE

Tabela 5: Gerenciamento de fichas no servidor.

ESTADOS EVENTOS	ACTIVE	WAIT-TOKEN
T-TOKEN.Req	p05 p07 [14] WAIT-TOKEN p09 p06 [24] [13] ACTIVE ^p06 [27] [15] [23] ^p07 [24]	
T-TOKEN-Rsp	p10 [22] [16] [12] p08 p06 [24] [13] ^p06 [27] [15] [23]	

TOKEN	p10 [22] [21] p08 [27] [15] [23]	p10 [22] [21] p08 p06 [24] [13] ^p06 [27] [15] [23]
PASS	p05 p07 [14] p09 [27] [15] [23] ^p07 [28]	
ERROR(*e-code* = 9)	[20] [22]	
ERROR(*e-code* =10)	[20] [22]	
Estouro TTIMER	p10 p12 [24] [22] [12] ^p12 [25] [17] [18]	
Estouro WTIMER	p11 p10 [25] [17] [19] ^p11 [22]	

4 - Implementação do Protocolo MCF

Alta velocidade, flexibilidade e modularidade são os principais objetivos da arquitetura de implementação desenvolvida pelo Grupo de Teleinformática e Automação (GTA) da UFRJ. Desde a primeira versão do sistema [Bagi91], tornou-se claro que uma implementação cuidadosa seria necessária, levando-se em consideração disponibilidades de memória e *overheads* introduzidos pelo sistema operacional. Portanto, uma arquitetura de implementação eficiente, com módulos especializados de gerenciamento de memória, escalonamento de tarefas e gerenciamento de temporizações, foi definida no sentido de se obter ganhos significativos no desempenho [Nune94a] [Nune95a] [Nune95b].

A Figura 4 mostra o lugar do protocolo MCF na arquitetura de implementação desenvolvida pelo GTA. Quatro módulos principais sobressaem nesta figura: o módulo **MAC**, padrão IEEE 802.3, o módulo **MCF**, o módulo **Usuário**, que implementa as rotinas de teste e análise de desempenho do protocolo, e o módulo

Globais, que compreende os módulos Gerente de Memória, Escalonador de Tarefas e Gerente de Temporizações.

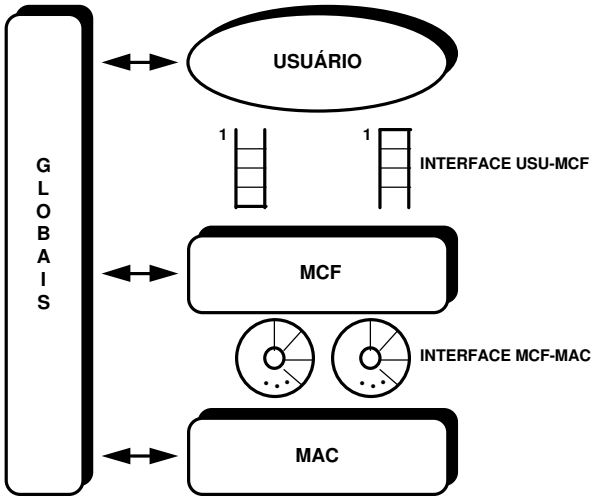

Figura 4: Sistema implementado.

5 - Análise de desempenho

O sistema implementado está codificado em linguagem C e roda em computadores PC-compatíveis sob o sistema operacional DOS. As mais de 8400 linhas de código geram um programa executável de aproximadamente 170 koctetos, com informação de depuração. O ambiente de medidas consiste em PCs equipados com CPUs 486 DX, conectados a uma rede local Ethernet através de placas padrão NE2000.

Esta análise de desempenho consiste fundamentalmente em medidas de **vazão** para um usuário sobre a camada MCF, em diferentes configurações. Doravante, vazão significará o total de dados do usuário (cabeçalhos ou caudas de PDUs não são levados em consideração) transmitido em uma unidade de tempo.

O ideal seria que o tempo de transmissão fosse medido a partir do instante em que o transmissor envia a primeira mensagem até a recepção da última mensagem pela entidade remota. Entretanto, isto exigiria um sistema de medidas distribuído. Outra metodologia foi então adotada no usuário de medidas: o transmissor computa o intervalo de tempo decorrido entre o envio da primeira mensagem e a recepção do reconhecimento da última mensagem. Esta medida utiliza as interrupções do relógio do PC, geradas a cada 55 ms. Portanto, uma quantidade mínima de 200 quadros é transmitida para manter o erro de medida em um limite tolerável. Nenhuma medida de estabelecimento ou liberação de conexões foi realizada, em virtude da necessidade de interface com o usuário para a determinação das opções da conexão. Pode-se dizer também que os computadores eram dedicados à transferência de dados, em um horário de pouco tráfego na rede. O usuário realizava apenas transferências de memória para memória, evitando assim qualquer tipo de retardo introduzido por dispositivos de

entrada e saída. As operações de exibição no vídeo foram inclusive reduzidas a um mínimo necessário durante a transferência propriamente dita.

As medidas incluem o *overhead* introduzido pelo cabeçalho/cauda da camada MCF e pelo cabeçalho/cauda da camada MAC, em um total de 58 octetos por mensagem. Na verdade, essas medidas de vazão são limites inferiores pois incluem o processamento do usuário na criação/destruição de mensagens, o processamento dos dados nas interfaces, alocações e dealocações de memória, etc.

Foi observada uma perda de aproximadamente 37%, pelo protocolo MCF com nenhum mecanismo opcional habilitado (vazão de aproximadamente 3,80 Mbps), em relação ao desempenho obtido por um usuário da camada MAC (vazão de aproximadamente 6,20 Mbps). Esta perda é causada principalmente pela inevitável cópia dos dados em uma região contígua de memória para atender às exigências dos controladores de comunicação. Mesmo assim, uma vazão de 1,28 Mbps é conseguida para uma conexão ponto-a-ponto em uma configuração com *checksum*, controle de fluxo por janela e recuperação de erros por retransmissão seletiva, ou seja, o máximo *overhead* de processamento possível.

Quanto à eficiência do mecanismo de controle de fluxo através do controle da taxa de transmissão, observou-se uma variação percentual média de apenas 4% entre valores práticos e teóricos.

Quanto às suas características multidestinatárias, o algoritmo *bucket* permite que grupos arbitrariamente grandes sejam formados. Uma vazão de aproximadamente 1,36 Mbps é conseguida com o algoritmo *bucket* em um grupo de seis participantes. O mecanismo de fichas funcionou corretamente para duas fichas, sendo que este número pode ser facilmente aumentado, permitindo assim que um número arbitrário de membros possa transmitir simultaneamente seus dados para o grupo.

6 - Conclusões

Neste trabalho foi apresentado um protocolo multiponto-a-multiponto com configuração dos parâmetros da comunicação e com gerenciamento de acesso baseado em fichas, o protocolo MCF. Com a popularização dos serviços multimídia, há um sentimento generalizado de que os próximos protocolos deverão ser adaptáveis aos diferentes requisitos de desempenho de cada aplicação. Neste sentido, o protocolo MCF oferece a possibilidade de seleção de uma gama de mecanismos:

- Cálculo e verificação de *checksum* opcionais sobre o campo de dados do usuário em PDUs DATA.
- Dois mecanismos de controle de fluxo: janelas de transmissão e controle de taxa.
- Dois mecanismos de recuperação de erros: *go-back-N* e retransmissão seletiva.
- Algoritmo *bucket* opcional para comunicações multidestinatárias.

Assim, uma aplicação de transferência de arquivos pode optar por comprometer um pouco seu desempenho em troca de maior confiabilidade para os dados transmitidos, acionando o cálculo de *checksum*, o controle de fluxo e a recuperação de erros, enquanto um serviço de datagrama mais preocupado com o aproveitamento

máximo da banda-passante disponível pode optar por não acionar nenhum mecanismo opcional. Por outro lado, o controle de fluxo através do controle da taxa de transmissão mostra-se adequado à aplicações de tempo real em que rígidos requisitos de banda-passante e atraso se fazem necessários.

No que tange às comunicações multidestinatárias, o algoritmo *bucket* implica em cálculo de *checksum* e em recuperação de erros por *go-back-N*, apresentando um desempenho ainda melhor que o de uma conexão ponto-a-ponto com configuração semelhante. Isso mostra que o algoritmo *bucket* possibilita uma maior continuidade no fluxo da transmissão, quando comparado ao controle por janela. Além disso, o caráter notadamente cliente/servidor da implementação, com um servidor de fichas de endereço fixo, permite uma constante alteração no tamanho do grupo, sem afetar o andamento da comunicação.

Todo o protocolo segue a arquitetura de implementação desenvolvida pelo GTA, cujos principais méritos são os seguintes:

- comunicação assíncrona entre camadas, modelada por filas FIFO. O resultado é um alto grau de modularidade e um fraco acoplamento entre as camadas;
- estruturas de dados padronizadas nas interfaces, implementadas com ponteiros para evitar cópias na transferência de dados de uma camada para outra e nos procedimentos de segmentação e retransmissão;
- um gerenciamento de memória eficiente com processamento diferenciado para a alocação / liberação de *buffers* de tamanho fixo e variável, separando regiões de memória independentes para transmissão e recepção e reduzindo a fragmentação ao mínimo possível;
- um escalonamento de tarefas que evita testes de filas desnecessários e se aproveita da atomicidade de algumas funções;
- um gerenciamento de temporizações flexível em que estouros são tratados como primitivas de serviço.

As medidas efetuadas permitiram uma análise apurada da eficiência de cada mecanismo e comprovaram a alta flexibilidade e desempenho do protocolo implementado.

Atualmente este protocolo está sendo portado para uma plataforma SUN-SPARC com sistema operacional UNIX. Encontra-se em andamento projetos de um escalonador de tarefas com prioridade (exigido por aplicações com restrições temporais), interfaceamento com o protocolo MBONE, aplicação de conferência usando Tcl/tk e testes com uma camada de sincronização multimídia.

7 - Referências

[Arms92] S. Armstrong, A. Freier e K. Marzullo, “Multicast Transport Protocol”, RFC 1301, Network Information Center, SRI International, fevereiro de 1992.

[Bagi91] L.F. Baginski, F.M.C. de Barros, R. Schatzmayr e O.C.M.B. Duarte, "Implementação e Análise de Desempenho em um Sistema de Transferência de Dados", *X Congresso da Sociedade Brasileira de Computação*, Vitória, Brasil, pp. 157-169, julho de 1991.

[Brau93] R. Braudes e S. Zabele, "Requirements for Multicast Protocols", RFC 1458, Network Information Center, SRI International, maio de 1993.

[Cher89] 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.

[Ches79] G. Chesson, "Datakit Software Architecture", *Proceedings of the ICC*, pp. 20.2.1-20.2.5, 1979.

[Clar87] 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.

[Doer90] W. Doering et al, "A Survey of Light-weight Transport Protocol for High Speed Networks", IEEE Transactions on Communications, vol. COM-38, n. 11, pp. 2057-2071, novembro de 1990.

[Nune94a] M.D. Nunes, C.V.N. Albuquerque e O.C.M.B. Duarte, "Performance Measurements in a Manufacturing Communication System", *ISCAS '94*, Londres, maio de 1994.

[Nune94b] M.D. Nunes e O.C.M.B. Duarte, "Análise de Mecanismos para Protocolos de Alto Desempenho", *VI Simpósio Brasileiro de Arquitetura de Computadores e Computação de Alto Desempenho*, Caxambu, agosto de 1994.

[Nune95a] M.D. Nunes, "Protocolo Multidestinatário com Configuração de Facilidades e Gerenciamento de Acesso", *Tese de Mestrado*, PEE-COPPE/UFRJ, abril de 1995.

[Nune95b] M.D. Nunes, C.V.N. Albuquerque e O.C.M.B. Duarte, "An Efficient Implementation Architecture for Layered Communication Systems", *38th Midwest Symposium on Circuits and Systems*, Rio de Janeiro, Brasil, 13 à 16 de agosto de 1995.

[Nune95c] M.D. Nunes e O.C.M.B. Duarte, "Proposta e Implementação de um Protocolo Multidestinatário Configurável", *VII Simpósio Brasileiro de Arquitetura de Computadores - Processamento de Alto Desempenho,* pp. 229-242, Canela, RS. julho de 1995.

[PEI92] Protocol Engines Inc., "Xpress Transfer Protocol Definition", Revisão 3.6, janeiro de 1992.

[Sant92] H. Santoso e S. Fdida, "Transport Layer Multicast: XTP Bucket Error Control and Its Enhancement", *4th. IFIP Conference on High Performance Networking*, Liège, Bélgica, 16-18 de dezembro de 1992.

[Topo90] C. Topolcic, "Experimental Internet Stream Protocol", RFC 1190, Network Information Center, SRI International, outubro de 1990.

[Wats89] 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.

[Zhan93] L. Zhang et al., "A New Resource ReSerVation Protocol", IEEE Network Magazine, vol.6, n.5, pp. 8-18, setembro de 1993.