Capítulo VI

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 dos mecanismos que eram outrora inerentes aos protocolos de transporte convencionais:

- 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. Apesar de já apresentar um bom desempenho, a precisão do controle de fluxo por taxa pode ser melhorada mediante a programação direta do controlador de temporizações do PC, reduzindo o intervalo do *timer tick*.

Por maior que possa parecer uma perda de aproximadamente 37% em relação ao desempenho obtido por um usuário da camada MAC, 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 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. Este algoritmo 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. 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. Entretanto, a maneira como as aplicações administrarão os dados recebidos de diferentes fontes foge ao escopo deste trabalho.

Mas para tudo isso funcionar a contento, teve papel fundamental 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.

O protocolo MCF é um pequeno passo na direção de um futuro sistema integrado para aplicações multimídia a ser desenvolvido pelo GTA. Algumas extensões podem ser efetuadas, como por exemplo:

- implementação de mecanismos de roteamento, para permitir a interligação de diferentes redes locais através do protocolo MCF;

- portar o protocolo para estações de trabalho de alto desempenho com sistema operacional UNIX. Esta é uma das mais importantes sugestões de trabalhos futuros, pois sendo estas estações normalmente máquinas de 32 bites (diferentemente dos PCs disponíveis, em que apenas a CPU é de 32 bites), espera-se que o protocolo MCF apresente um melhor desempenho devido à sua característica de alinhamento em 32 bites. Além disso, o sistema operacional UNIX "enxerga" memória plana, não paginada como no DOS. Isso elimita um importante fator de limitação do PC, possibilitando o aumento dos *buffers* internos do MCF e o conseqüente aumento da janela de transmissão. Em um ambiente de estação de trabalho que ofereça um intervalo de temporização menor, pode-se ainda aumentar a precisão do controle de fluxo por taxa;

- acesso à Internet através do protocolo MCF. Para isso, bastaria desenvolver uma espécie de "camada MAC" sobre a interface soquete. Dessa forma, o protocolo MCF se posicionaria logicamente sobre a camada IP, através da escolha apropriada do soquete (*socket_raw*). No caso de uma camada IP Multiponto, conexões multidestinatárias poderiam ser implementadas sem maiores problemas;

- finalmente, ainda na plataforma de estação de trabalho com ambiente UNIX, outra possibilidade seria a implementação do protocolo MCF sobre os dois meios disponíveis de acesso à interface Ethernet: acesso direto e através do mecanismo de STREAMS [Luis95]. O acesso direto permite a utilização das interfaces padrão e dos módulos de gerenciamento de memória e escalonamento de tarefas desenvolvidos pelo GTA. O mecanismo de STREAMS, por sua vez, permite a implementação de um sistema multicamadas, pois possui sua própria interface, também baseda em filas FIFO, bem como seus próprios módulos de gerenciamento de memória e escalonamento de tarefas. Esta seria uma boa oportunidade de comparar o desempenho da arquitetura de implementação desenvolvida pelo GTA com o mecanismo de STREAMS.