UMA IMPLEMENTAÇÃO DAS CAMADAS MAC/LLC PARA A INTERFACE MINI-MAP*

David J. Turnell
Joberto, S.B.M.

GRC/UFPb - Grupo de Redes de Computadores
Instituto de Tecnologia Eletro-Eletrônica (ITEEL)
CP 10117
58100 - Campina Grande - Pb

SUMÁRIO

Este artigo descreve uma implementação das sub-camadas MAC ("Media Access Control") e LLC ("Logical Link Control") tipo 3 para uma rede industrial com arquitetura mini-MAP. Esta implementação é parte do projeto de um processador frontal de comunicação ("front-end") de rede. Este artigo aborda os seguintes aspectos da implementação: o ambiente de desenvolvimento; a estruturação da implementação do software MAC/LLC; a utilização dos recursos do processador frontal; e a facilidades de teste das camadas implementadas.

1. INTRODUÇÃO

A rede local MAP ("Manufacturing Automation Protocol") [1] foi especificada originalmente pela General Motors para formar a espinha dorsal de sistemas de controle industrial de grande porte. O modelo ISO/OSI ("International Standards Organization - Open Systems Interconnection") [2] foi usado como base de sua especificação.

A rede MAP responde aos requisitos de comunicação nos níveis superiores da hierarquia de sistemas industriais [3], utilizando um meio físico banda-larga e implementações de todas as sete camadas do modelo OSI. A arquitetura MAP "backbone" não é apropriada para os níveis inferiores da hierarquia em função do alto custo e complexidade da interface e do tempo de resposta excessivo para controle em tempo real [4].

Para melhorar o desempenho da arquitetura MAP em aplicações de tempo real, e também para criar interfaces mais simples (para sensores, por exemplo), a versão MAP 3.0 introduziu a alternativa mini-MAP. A mini-MAP utiliza um meio físico mono-banda e utiliza o equivalente às camadas 1,2 e 7 do modelo OSI. A figura 1 mostra a arquitetura de protocolos de uma interface mini-MAP.

---

* Trabalho desenvolvido com o apoio do CPqD/Telebrás

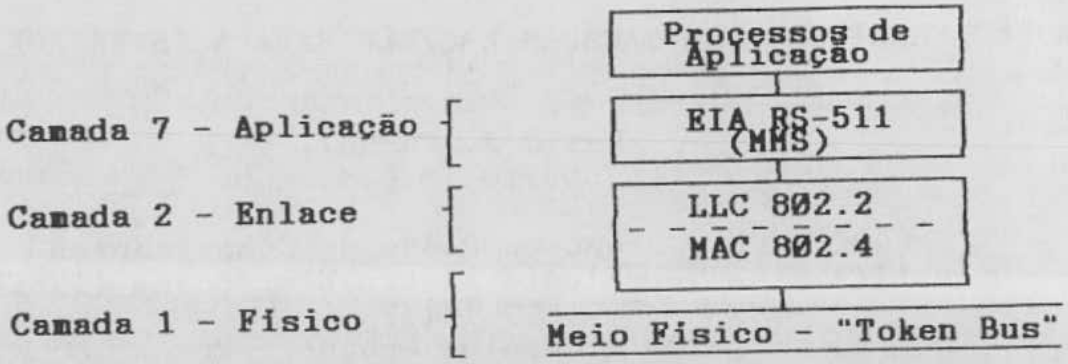

Figura 1: Protocolos da Interface mini-MAP

A primeira camada mini-MAP é a MAC ("Media Access Control"). A MAC da mini-MAP é um barramento com ficha ("Token Bus") definida pela IEEE como padrão 802.4. Nesta arquitetura de rede as estações são ligadas em um anel lógico, em que circula uma única ficha a qual dá o direito de transmissão [5].

A segunda camada mini-MAP é a LLC tipo 3 ("Logical Link Control - Acknowledged Connectionless Service") definida pela IEEE como padrão 802.2 [6]. A LLC tipo 3 é do tipo 'um quadro, um reconhecimento', e é baseada na operação RWR ("Request with Response"), onde cada quadro transmitido recebe um quadro de reconhecimento, que pode ou não conter dados.

A camada superior da arquitetura mini-MAP é o EIA RS-511, que corresponde à camada de aplicação do modelo OSI/ISO.

Este artigo descreve uma implementação das camadas MAC e LLC tipo 3 segundo a especificação ISO [7]. Este trabalho é parte do projeto de um Processador Frontal de Comunicação (PFC) para a rede mini-MAP. Nesta primeira fase do projeto o PFC integra um hardware, um núcleo multi-tarefa, e as camadas MAC e LLC.

2. O PROCESSADOR FRONTAL DE REDE

Esta seção descreve o Processador Frontal de Comunicação (PFC) em que se situam as camadas MAC e LLC desta implementação. O PFC é um "front-end" que reduz a carga na CPU e na memória imposta no hospedeiro pela interface de rede. O PFC é uma interface inteligente que permite o deslocamento das três camadas da mini-MAP para fora do hospedeiro.

O hospedeiro do PFC é o PP (Processador Preferencial) [8] desenvolvido pela Telebrás, uma máquina da classe super-micro projetada para diversas ambientes, inclusive para o ambiente industrial. O PFC tem três componentes principais: o hardware, o executivo multi-tarefa, e as tarefas que executam as camadas do

protocolo da interface de rede.

O hardware do PFC utiliza o controlador de rede MC68824 compatível com o padrão IEEE 802.4, é baseado no microprocessador 80186 e pode dispor de uma capacidade de memória de 512 Kbytes ou 1 Mbyte.

O executivo é um núcleo multi-tarefa otimizado para operação em tempo real [9]. Ele fornece servicos de gerenciamento de memória, de comunicação inter-tarefa, de temporização, e de gerenciamento de tarefa. O escalonamento de tarefa é feito na forma preemptiva, com tempo na UCP proporcional à prioridade da tarefa.

As tarefas são as implementações das camadas da interface. No caso da arquitetura mini-MAP, o PFC terá no mínimo duas tarefas, a primeira executando as camadas MAC/LLC e a segunda a camada de aplicação (camada EIA RS-511) que não faz ainda parte do projeto PFC.

3. O AMBIENTE DE DESENVOLVIMENTO

Antes de começar a implementação das camadas, faz-se necessário definir o ambiente de desenvolvimento - aspectos como o ambiente do hospedeiro, a linguagem de programação, e o formato do código das tarefas.

3.1 O AMBIENTE DO HOSPEDEIRO

No projeto do PFC foram desenvolvidos vários software além daqueles das camadas MAC/LLC. Estes incluem um executivo, um carregador de tarefas, um "driver" do hospedeiro, e um conjunto de programas de teste.

O hospedeiro PP e o PFC são baseados na familia de microprocessadores Intel. Por razões de geração de código e de teste, este fato limitou a escolha do ambiente em dois sistemas operacionais compatíveis com esta familia - o XENIX e o MS-DOS. Os dois geram código 8086 e os dois são confiáveis. Embora o ambiente XENIX seja o melhor dos dois para o desenvolvimento, ele não estava disponível e, por isso, o MS-DOS foi adotado.

3.2 A LINGUAGEM

A linguagem usada na implementação da MAC/LLC foi em parte determinada por uma decisão inicial do projeto do PFC - a de usar uma única linguagem para seus vários componentes de software. A linguagen C foi utilizada por várias razões, entre elas:

- o código objeto é eficiente;

- a linguagem é estável e permite transportabilidade;
- os compiladores C para o MS-DOS são eficientes e
- existe uma experiência acumulada em programação C entre os participantes da equipe de desenvolvimento.

Para a prototipagem, o Microsoft QuickC foi usado em função da sua eficiência no ciclo de edição/compilação. Para a compilação final foi usado o Microsoft C 5.1 por causa de sua eficiência na otimização. Na programação não foram utilizadas as funções das bibliotecas C que dependem do ambiente MS-DOS, porque este ambiente não existe no PFC.

3.3 FORMATO DAS TAREFAS

O código objeto das tarefas é produzido no ambiente MS-DOS mas executa no ambiente do PFC. O carregador de tarefa que executa sob MS-DOS na inicialização do sistema recebe os códigos das tarefas na forma de arquivos e os instala no PFC. Isso introduziu a necessidade de formatar os arquivos adequadamente.

Para evitar o desenvolvimento de um "linker" especial, o formato do arquivo é aquele do arquivo EXEC do MS-DOS, ou seja, o que o compilador normalmente produz. A única modificação no processo normal de compilação foi a substituição do código de inicialização do ambiente MS-DOS, inserido pelo compilador por uma simples função de inicialização de tarefa, escrita em assembler e apropriado para o ambiente de execução no PFC.

4. IMPLEMENTAÇÃO DA MAC

A especificação ISO define três primitivas na interface entre as camadas LLC e MAC:

MA_UNITDATA_request;
MA_UNITDATA_indication e
MA_UNITDATA_STATUS_indication.

A especificação da LLC declara na sua seção 2.2 que a definição destas primitivas não implica numa interface "exposta" entre as duas camadas. No PFC isso significa que não há necessidade de implementar a MAC como uma tarefa separada da LLC. Para o PFC as primitivas MAC foram implementadas na forma de funções internas da LLC.

Esta flexibilidade é necessária devido ao aumento contínuo na capacidade dos controladores de rede. Além de uma boa parte da camada MAC, os controladores atuais fazem algumas partes da LLC que são críticas no tempo. Por exemplo, o MC68824 automaticamente envia um reconhecimento de um quadro recebido sem necessitar da intervenção do software da LLC. Esta interação violaria uma interface "exposta" entre estas duas camadas.

Para o PFC cada uma das três primitivas foi implementada como uma função para ser chamada pela LLC. A implementação destas funções é totalmente dependente das minúcias de funcionamento do MC68824, sobre as quais não há espaço neste artigo para uma abordagem profunda.

4.1 O FORMATO DO PARâMETRO `DATA´

As três primitivas têm como um dos seus parâmetros o `Data´. Nesta implementação este parâmetro tem a seguinte estrutura:

```
struct DATA {
     char *data_ptr;
     int length;
     int offset;
};                          /* estrutura tipo DATA */
```

O `data_ptr´ contém o endereço do início do buffer de dados. O `offset´ indica o número de bytes entre o início do buffer e o começo dos dados. O `length´ contém o número de bytes de dados a partir do início do campo de dados.

O `offset´ deixa espaço para os três campos usados na construção de uma MSDU ("MAC Service Data Unit"). Estes campos são `frame_control´, `destination_address´, e `source_address´. Este espaço tem que ser no mínimo de 13 bytes. O tamanho máximo dos dados mais os três campos é de 8 Kbytes.

4.2 DESCRITORES DE QUADRO E DE BUFFER

Para que uma MSDU possa ser manipulada pelo MC68824, dois tipos de descritores têm que ser criados. O primeiro descritor usado pelo MC68824 é o "Buffer Descriptor" (BD). Este contém informação sobre um buffer dos dados que compõe o quadro. Podem haver vários BD´s (cada um com seu buffer) por quadro. O segundo descritor é o "Frame Descriptor" (FD), que contém informação sobre o quadro inteiro (endereço do primeiro BD, status, tamanho, etc ...).

No caso de transmissão, a MSDU entregue à MAC pela LLC é contígua e há necessidade para somente um BD. No caso de recepção, podem haver vários BD´s por quadro porque os buffers na lista de recepção são de tamanho fixo (2 kbytes). Quando isso acontece, a MAC cria um buffer de tamanho suficiente e copia nele os conteúdos dos varios BD´s para que possa entregar uma LSDU contígua à LLC. Isto impõe uma sobre carga para quadros de tamanho superior a 2 kbytes.

4.3 A PRIMITIVA MA_UNITDATA_request

A função que implementa esta primitiva tem a seguinte definição:

```
ma_unitdata_request(source,destination,data,priority,
                                                class)

char source[6],
char destination[6],
struct DATA data,
int  priority,
int class);
```

Os parâmetros `source´ e `destination´ são arranjos de seis bytes representando endereços. O `data´ representa a LSDU ("Link Service Data Unit") a ser transmitida.

Primeiro, a LSDU é convertida numa MSDU através da instalação dos três campos mencionados na seção 4.1. Em seguida, um FD e um BD são criados para a MSDU. Finalmente, o FD é inserido na fila de transmissão correspondente à sua prioridade.

4.4 A PRIMITIVA MA_UNITDATA_indication

A função que implementa esta primitiva tem a seguinte definição:

```
int ma_unitdata_indication(source,destination,data,
                                reception_status,priority,
                                        service_class)

char *source[],
char *destination[],
struct DATA *data,
int *reception_status,
int *priority,
int *service_class
```

Esta função passa à LLC uma LSDU recebida pela MAC. A LLC chama esta função e recebe de volta um `1´ se houver quadro recebido e um `Ø´ caso contrário. Se hóuver quadro, esta função utiliza os apontadores passados como parâmetros para passar à LLC o quadro recebido.

4.5 A PRIMITIVA MA_UNITDATA_STATUS_indication

A função que implementa esta primitiva tem a seguinte definição:

```
int ma_unitdata_status_indication (source,destination,
                        transmission_status,priority,
                                        service_class)

char *source[],
```

```
        char *destination[],
        int *transmission_status,
        int *priority,
        int *service_class
```

Esta primitiva indica à LLC o estado ("status") de uma primitiva ma_unitdata_request lançada anteriormente. Como no caso da ma_unitdata_indication, a função devolve um `1´ se há um estado para relatar e `Ø´ caso contrário.

5. A IMPLEMENTAÇÃO DA LLC

Esta seção descreve a implementação da camada LLC.

5.1 A COMUNICAÇÃO ENTRE AS CAMADAS

No PFC as camadas são implementadas na forma de tarefas, uma delas sendo a MAC/LLC com interface interna. A LLC tem uma interface "exposta" com a camada superior, que no caso da arquitetura mini-MAP é uma tarefa executando a camada EIA RS-511.

Esta comunicação é feita com o auxílio dos serviços do executivo ligados à passagem de mensagens entre tarefas:

```
        x_send_message e
        x_receive_message
```

Como exemplo, suponha que a LLC queira mandar uma primitiva DL_DATA_ACK_indication para o EIA RS-511. De início, ela preencheria uma estrutura DL_DATA_ACK_MESSAGE que tem o seguinte formato:

```
struct DL_DATA_ACK_MESSAGE {
        int message_type;
        char source_address[6];
        char destination_address[6];
        struct DATA data;
        int priority;
        int service_class;
};
```

Depois de preencher uma destas estruturas, a LLC utilizaria o serviço `x_send_message´ do executivo para mandar esta estrutura à tarefa da camada RS-511. O executivo se encarregaria de armazenar a mensagem até que a tarefa RS-511 peça sua recepção com a primitiva `x_receive_message´.

5.2 O GERENCIAMENTO DOS BUFFERS

O MC68824 mantém uma lista ("pool") de buffers de recepção,

cada buffer tendo um tamanho de 2 kbytes. Na recepção de um quadro, o MC68824 retira buffers da lista, formando uma cadeia de buffers por quadro (a 802.4 define que o tamanho máximo de quadro é de 8 kbytes). A LLC precisa preencher esta lista na inicialização do sistema com a quantidade POOL_SIZE de buffers. A memória para cada buffer é obtida do executivo através do serviço `x_request_mem´.

Durante sua operação, o MC68824 pode avisar, através de uma interrupção, que a quantidade de buffers na lista está baixa. Então, a LLC deve criar mais buffers com `x_request_mem´. A LLC mantém uma contagem do número de buffers na lista. Quando o número ficar acima de POOL_SIZE, os buffers de recepção já processados não serão mais colocados de volta na lista mas destruidos e sua memória devolvida ao executivo com o serviço `x_return_mem´.

5.3 VARIáVEIS DE ESTADO

A camada LLC é responsável pela manutenção de três tipos de variável de estado - V(SI), V(RI) e V(RB). A implementação das três é idêntica com a descrição que segue para V(SI) tambem se aplicando as outras duas.

V(SI) é uma variável com dois possíveis valores - 0 ou 1. O valor em qualquer instante é igual ao "code-point" recebido na última LSDU. A LLC mantém um V(SI) para cada combinação de endereço de destino (DA), ponto de acesso local (SSAP) e prioridade (P).

A função que devolve o valor de V(SI) usa o DA para ter acesso a uma tabela de "hash", que contém um apontador para o primeiro de uma cadeia de elementos. Cada elemento é uma estrutura que representa um V(SI). A estrutura é a seguinte:

```
struct VSI {
        int da[3];
        char source_service_access_point;
        char priority;
        char vsi;
        int timer;
        struct VSI *next;
};
```

Usando o apontador `next´, a LLC segue a cadeia de elementos até chegar no elemento que combina com o DA, SSAP e P. Se o elemento não estiver presente na cadeia, a LLC pede um bloco de memória do executivo (com x_request_mem) e instala um novo elemento no início da cadeia. O encadeamento é dinâmico, ou seja, cada elemento localizado é colocado no início da cadeia (na tabela "hash"). A cadeia tem então os elementos mais referênciados no seu início, reduzindo o tempo médio de busca.

Cada elemento contém o índice de um temporizador. Quando o elemento é criado, um temporizador é também criado (x_set_timer). Cada vez que uma V(SI) é atualizada, o temporizador é reiniciado. Se o temporizador conseguir terminar ("time-out"), a V(SI) correspondente pode ser eliminada da cadeia e sua memória devolvida ao executivo.

5.4 A ESTRUTURAÇÃO DA LLC

A sub-seção seguinte descreve a estrutura geral do software da LLC.

5.4.1 A Função Principal

A LLC foi implementada na forma de um laço em que três atividades principais são processadas sequencialmente: a recepção de mensagens; o teste do estado de transmissão; e o teste do estado de recepção. A funcão principal ("main") da camada LLC é a seguinte (em pseudo-código):

```
main
{
     initialize_llc;
loop:
     check_messages;
     check_tx_status;
     check_rx_status;
     goto loop;
}
```

Uma vez iniciada, a tarefa MAC/LLC executa indefinidamente no laço principal. A etapa de "bootstrap" da tarefa é a seguinte:

- a tarefa é instalada no PFC pelo carregador de tarefa;
- o executivo começa sua execução;
- a primeira vez que a tarefa recebe tempo na UCP, a execução começa com a rotina de inicialização inserida pelo compilador e
- esta rotina de inicialização chama a função `main´ acima.

5.4.2 A Função `check_messages´

Esta função obtem mensagens na fila de entrada. Ela usa o serviço `x_receive_message´ do executivo. Se houver mensagem, ela chama uma função que cuida da mensagem. O pseudo-codigo da função é o seguinte:

```
check_messages
{
```

```
            if x_receive_message(message) = true
                switch (message->type) {
                    case  DL_DATA_ACK_request
                          dl_data_ack_request()
                    case  DL_REPLY_request
                          dl_reply_request()
                    case  DL_REPLY_UPDATE_request
                          dl_reply_update_request()
                }
            }
        }
```

Os "cases" processam as três primitivas que a LLC recebe da EIA RS-511 através da chamada de uma função de serviço.

5.4.3 A Função `check_tx_status´

Esta função testa as quatro filas de transmissão a procura de quadros que tenham sido transmitidos e reconhecidos pela estação receptora.

```
check_tx_status
{
    if ma_unitdata_status(source_address,
                destination_address,tx_status,priority,
                                service_class) = 1 {

        if no_acknowledgemente {
            if data_ack_service
                DL_DATA_ACK_status(unsuccessful)
            else
                DL_REPLY_STATUS_indication(unsuccessful)
        }
        else
            get V(SI)
            if V(SI) = codepoint {
                if data_ack service
                    DL_DATA_ACK_STATUS_indication(ok)
                else
                    DL_REPLY_STATUS_indication(ok)
                invert V(SI)
            }
        }
    }
}
```

5.4.4 A Função `check_rx_status´

Esta função testa as quatro filas de recepção a procura de quadros que tenham sido recebidos.

```
check_rx_status
{
     if ma_unitdata_indication(&source_address,
                         &destination_address,&data,
                         &reception_status,&priority_class,
                                        &service_class) = Ø {

          get V(RI)
          if V(RI) = codepoint {
               if data_ack service {
                    if data
                         L_DATA_ACK_indication
                    }
                    else {
                         if acknowledgement sent
                              L_REPLY_indication
                         else
                              if no data
                                   L_DATA_ACK_indication
                    }
               }
               return FD and BD descriptor memory
          }
     }
}
```

6. O TESTE DA MAC/LLC

O teste da MAC/LLC foi dificultado porque ele executa num ambiente fora do alcance das ferramentas de depuração normalmente usadas no hospedeiro (por exemplo: Debug ou CodeView). Dois métodos de teste são possíveis. No primeiro, o software poderia ter sido modificado para rodar no hospedeiro e depurado com CodeView. No segundo, o software poderia ter sido testado como uma caixa preta enquanto funcionava no PFC.

O problema do primeiro método é que o software tem uma dependência muito forte com o controlador MC68824. Este controlador é complexo demais para emular com rotinas de software. Assim,testar o software no hospedeiro não teria permitido a depuração da maioria da MAC, resultando na necessidade de usar depois o segundo método.

Para executar o teste mais eficientemente, o segundo método foi escolhido. O programa de teste executa no PP e se comunica com a LLC através de um "driver". Este "driver´ serve como interface entre o PFC e software de aplicação no PP. Em um dos testes o MC68824 é colocado no modo "loop-back" interno, e a LLC estimulada pelo programa de teste com a primitiva DL_DATA_ACK_request. O endereço destino dado é o `broadcast´, resultando na recepção automática dos dados e a geração da primitiva DL_DATA_ACK_indication. Testes similares são feitos

com as outras primitivas.

Para isolar os problemas são necessárias uma série de mensagens de estado enviadas através do executivo para o programa de teste. O que causa mais trabalho é a verificação de apontadores. Para ajudar nisso, o executivo foi acrescido de um serviço de verificação de apontador, que devolve o dono (número da tarefa) da memória para a qual o apontador aponta.

O teste final consiste em instalar dois PFC's no mesmo PP, e monitorar a validade das LSDU's trocadas nos dois sentidos.

7. CONSIDERAÇÕES FINAIS

O deslocamento de alguns serviços básicos para um executivo, e a organização das camadas na forma de tarefas permitiu uma implementação compacta e simplificada da camada MAC/LLC.

Na medida em que os controladores de rede aumentam suas capacidades e executam mais funções da camada LLC, sua interface formal com a camada MAC deixa de ser relevante. É mesmo possível prever-se a introdução de controladores de rede programáveis implementando totalmente ambas as camadas MAC e LLC.

A rede MAP é um elemento importante na automação industrial. Com o congelamento por alguns anos da especificação MAP 3.Ø, chegou-se a fase onde a implementação e disseminação do padrão são importantes. Esta implementação das camadas MAC/LLC faz parte de um esforço no sentido da obtenção de uma tecnologia de rede local para a automação industrial com a utilização de uma plataforma de hardware nacional, no caso o Processador Preferencial.

Pode-se considerar o projeto do PFC como o primeiro passo na direção de uma interface MAP completa. Neste sentido o hardware e o executivo foram projetados para permitir a instalação no PFC de todas as camadas MAP.

7. REFERÊNCIAS

[1] - General Motors; "MAP Specification :- Version 3.Ø", April, 1987.

[2] - ISO/IS 7498, "Information Processing Systems. - Open Systems Interconnection. - Basic Reference Model", October, 1985.

[3] - Roberto E. Leite e Manuel J. Mendes; "Protocolos de Aplicação em Redes Locais de Computadores na Automação

Industrial (Protocolos MAP/TOP)", 5º Simpósio Brasileiro de Redes de Computadores, Abril 1987.

[4] - W.Timothy Strayer and Alfred C.Weaver, "Performance Measurement of Data Transfer Services in MAP", IEEE Network, vol. 2, No. 3, May, 1988.

[5] - Institute of Electrical and Electronic Engineers; "IEEE 802.4 - Token Passing Bus Access Method and Physical Layer Specification", 1988.

[6] - Institute of Electrical and Electronic Engineers; "IEEE 802.2 - CSMA-CD Acess Method and Physical Layer Specification", 1988.

[7] - ISO/IEC JTC 1/SC 6 N 4960; "Standards for Local Area Networks: Logical Link Control - Type 3 Operation, Acknowledged Connectionless Service", Junho, 1988.

[8] - Telebrás - CPqD; "Documentação Técnica do Projeto PP (Processador Preferêncial)", 1987.

[9] - David J Turnell e Joberto S B Martins; "Executivo Multi-Tarefa para Processador Frontal de Rede", 7º Simpósio Brasileiro de Redes de Computadores, Março, 1989.