

Protocolo de Ligação de Dados

Relatório



Universidade do Porto

Faculdade de Engenharia

FEUP

Redes de Computadores

3º ano

Mestrado Integrado em Engenharia Informática e Computação

Turma 4

Carolina Moreira	201303494	up201303494@fe.up.pt
Daniel Fazeres	201502846	up201502846@fe.up.pt
José Peixoto	200603103	ei12134@fe.up.pt

7 de Novembro de 2016

Conteúdo

1	Introdução	1
2	Arquitetura	1
3	Estrutura do código	1
4	Casos de uso principais	1
5	Protocolo de ligação lógica	2
6	Protocolo de aplicação	2
6.1	Envio de um ficheiro	2
6.2	Receção de um ficheiro	3
7	Validação	4
8	Elementos de valorização	4
8.1	Seleção de parâmetros pelo utilizador	4
8.2	Implementação de REJ	4
8.3	Verificação da integridade dos dados pela Aplicação	5
9	Conclusões	5

Resumo

No âmbito da unidade curricular de Redes de Computadores, foi-nos proposto o desenvolvimento de uma aplicação que testasse um protocolo de ligação de dados criado, transferindo um ficheiro recorrendo à porta de série *RS-232*. O trabalho permitiu praticar conceitos teóricos no desenho de um protocolo de ligação de dados como o sincronismo e delimitação de tramas, controlo de erros, controlo de fluxo recurso a mecanismos de transparência de dados na transmissão assíncrona.

Findo o projeto, notou-se a importância dos mecanismos que asseguram tolerância a falhas fornecidos pela camada de ligação de dados, uma vez que a camada física não é realmente fiável.

1 Introdução

2 Arquitetura

3 Estrutura do código

4 Casos de uso principais

A utilização do programa divide-se em dois propósitos distintos: envio ou receção de um ficheiro. A fase de tratamento dos parâmetros opcionais passados pela linha de comandos é comum em ambos os casos e engloba a chamada de funções como: `parse_args`, `parse_serial_port_arg`, `parse_flags` e chamadas opcionais a outras funções consoante os parâmetros passados.

Envio de ficheiro

Após a interpretação dos parâmetros opcionais e a leitura do ficheiro, o programa invoca a função `send_file` da camada de aplicação para o envio de um ficheiro que por sua vez pede à camada de ligação de dados que estabeleça uma ligação pela porta de série na chamada à função `transmitter_connect` e transmita dados através da função `transmitter_write` e termine a ligação com a chamada `disconnect`.

A chamada `transmitter_connect` da camada da ligação de dados abre a porta de série envia a trama SET e recebe a trama UA. No envio de tramas que requerem confirmação de receção é usada a função `f_send_acked_frame`. Antes da chamada à função `transmitter_write`, o programa organiza a informação a enviar sob a forma de pacotes de dados nas funções `send_control_packet` ou `send_data_packets`.

Receção de ficheiro

Após a interpretação dos parâmetros opcionais e a leitura do ficheiro, o programa invoca a função `receive_file` da camada de aplicação para a receção de um ficheiro que por sua vez pede à camada de ligação de dados que estabeleça uma ligação pela porta de série com a chamada à função `receiver_listen` e receba pacotes de dados através da função `receiver_read` e termine a ligação com a chamada `disconnect`.

A chamada `receiver_listen` da camada da ligação de dados abre a porta de série e espera pela receção de uma trama SET enviando em seguida a confirmação de receção com a função `f_send_frame`. Na camada de ligação de dados é usada a função `f_receive_frame` na receção de tramas. Após a receção de um pacote de dados através da função `llread`, o programa descodifica os dados recebidos nas funções `parse_control_packet` ou `parse_data_packet` caso se espere receber um pacote de controlo ou de dados respectivamente.

5 Protocolo de ligação lógica

6 Protocolo de aplicação

A camada de aplicação é responsável pela leitura/escrita dos dados do ficheiro a enviar/receber. Do lado do emissor, procede-se à segmentação do ficheiro em pacotes de dados que vão sendo numerados e enviados para a camada de ligação de dados, por forma a serem encaixados em tramas de informação e posteriormente enviados através da porta de série. Do lado do receptor, é feita a compilação e escritura dos dados recebidos num ficheiro em disco nomeado de acordo com a informação recebida nos pacotes de controlo. Quer no emissor quer no receptor, recorre-se à codificação das etapas sob a forma de máquinas de estado.

6.1 Envio de um ficheiro

A camada de aplicação pode interpretar os argumentos opcionais passados através da interface de linha de comandos para ler do disco um ficheiro para uma estrutura de dados que armazena os dados, o nome e o tamanho do ficheiro. Opcionalmente, só os dados de um ficheiro serão lidos do `stdin` para a estrutura de dados referida e será atribuído um nome de ficheiro predefinido.

```
struct file {  
    const char* name;  
    size_t size;  
    char* data;  
};
```

Após a leitura do ficheiro, a camada de aplicação entra numa máquina de estados com quatro estados ordenados: abertura de ligação, envio de pacote de controlo inicial, envio de pacotes de dados, envio de pacote de controlo final e fecho da ligação.

Método usado na abertura de ligação

```
int llopen(char *port, int transmitter);
```

Método usado para o envio de pacotes de controlo inicial e final

```
int send_control_packet(struct connection* connection,
    struct file *file,
    byte control_field);
```

Método usado para o envio de pacote de dados

```
int send_data_packets(struct connection* connection, struct
    file* file,
    size_t* num_data_bytes_sent, size_t* sequence_number
    );
```

Método usado para o envio dos pacotes para a camada de ligação de dados

```
int transmitter_write(struct connection* conn, byte*
    data_packet, size_t size);
```

Método usado no fecho da ligação

```
int llclose(const int fd);
```

6.2 Receção de um ficheiro

Após a interpretação dos parâmetros passados pela linha de comandos que indicam ao programa para receber um ficheiro, a camada de aplicação entra numa máquina de estados com quatro estados ordenados: abertura de ligação, receção de pacote de controlo inicial, receção de pacotes de dados e fecho da ligação. Após o estabelecimento de uma ligação com sucesso, o programa fica à espera da receção de um pacote de controlo com os dados relativos ao tamanho e nome do ficheiro. Posteriormente, inicia-se o processo de receção dos pacotes de dados com a informação contida no ficheiro até que se receba um pacote de controlo final, sinalizando o fim da receção do ficheiro.

Método usado para receber o pacote de controlo inicial

```
int receive_start_control_packet(const int fd,
    char **file_name, size_t *file_size)
```

Método usado para decodificar um pacote de controlo

```
int parse_control_packet(const int control_packet_length,
                        byte *control_packet, char **file_name,
                        size_t *file_size)
```

Método usado para receber pacotes de dados e escrever em disco

```
int receive_data_packets(const int fd, char* file_name,
                        size_t file_size, int attempts_left)
```

Método usado para decodificar um pacote pacote de dados

```
int parse_data_packet(const int data_packet_length,
                    byte *data_packet, char **data,
                    size_t* sequence_number)
```

Método usado para decodificar um pacote pacote de dados

```
int parse_data_packet(const int data_packet_length,
                    byte *data_packet, char **data,
                    size_t* sequence_number)
```

7 Validação

8 Elementos de valorização

8.1 Selecção de parâmetros pelo utilizador

Quando o programa é invocado pela linha de comandos de forma errónea ou sem quaisquer parâmetros adicionais, são mostrados os argumentos opcionais disponíveis que permitem configurar a execução do programa, nomeadamente o modo de operação (**receiver** ou **transmitter**), leitura do ficheiro a enviar do disco ou proveniente de dados redireccionados do **stdin**, selecção da baudrate, determinação do tamanho máximo de bytes de dados enviados em cada frame e do número de tentativas na recuperação de erros.

8.2 Implementação de REJ

Quando ocorrem erros no processamento de tramas recebidas na camada de ligação de dados, é enviada de forma preemptiva ao **timeout** uma trama com confirmação negativa (REJ) que permite a retransmissão da trama de informação.

```
else if (ret == BADFRAME_CODE) {
    /*
     * Send 'bad frame' acknowledgment.
     */
    byte c_out = data_reply_byte(conn->frame_number,
                                FALSE);
    if (f_send_frame(conn->fd, FRAME(c_out)) !=
        SUCCESS_CODE)
        break;
}
```

8.3 Verificação da integridade dos dados pela Aplicação

Após receção com sucesso do primeiro pacote de controlo pela camada de aplicação, é armazenado o tamanho expectável em bytes do ficheiro a receber, comparando-o no fim da sessão com o valor real de bytes dados recebidos. São também guardados os números de pacotes de dados perdidos e duplicados.

```
void receiver_stats()
{
    fprintf(stdout, "Receiver statistics\n");
    fprintf(stdout, "\treceived file bytes/file bytes:%zu/%zu\n",
            received_file_bytes, real_file_bytes);
    fprintf(stdout, "\tlost packets:%zu\n", lost_packets);
    fprintf(stdout, "\tduplicated packets:%zu\n",
            duplicated_packets);
}
```

9 Conclusões

Referências

- [1] Andrew S. Tanenbaum, David J. Wetherall, *Computer Networks*, Prentice Hall, 5th edition, 2011.