Universidade do Minho
Escola de Engenharia

Mestrado Integrado em Engenharia de Telecomunicações e Informática

Laboratórios de Telecomunicações e Informática I

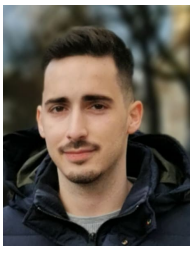
Desenvolvimento de uma aplicação de *chat*

Especificação da Fase 3

Grupo 4

Bruno Oliveira
A81570

Filipe Brás
A81307



João Cunha
A76645

José Bravo
A80132

Guimarães, 5 de dezembro de 2020

Índice

Lista de Figuras

1. Síntese da Fase 3

No âmbito da Unidade Curricular de **Laboratórios de Telecomunicações e Informática I** foi proposto a realização de um projeto prático, que tem como objetivo final o desenvolvimento e a implementação de uma aplicação de *chat*. Este relatório incide, apenas na especificação da fase 3 do projeto prático.

A fase 3 do projeto prático tem uma só tarefa, a tarefa 3.1 que tem como objetivo a implementação do controlo da ligação lógica. Esta fase irá incidir na camada 2 do modelo OSI, fornecendo interfaces API às camadas superiores.

Para tal, será necessário, estruturar a trama e os seus campos, definir o protocolo de comunicação, os tipos de trama e os respetivos campos e, finalmente, efetuar a transferência fiável de um fluxo de dados entre as placas ESP32, sobre a ligação RF sem fios, e as demais avaliações de desempenho do sistema teóricas e práticas de forma semelhante aos realizados na fase anterior.

Por último, um correto planeamento de tarefas é essencial, para que haja uma melhor distribuição de tarefas pelos elementos do grupo, organização e eficácia.

2. Tarefa 3.1 - Controlo da ligação lógica

2.1. Estrutura da Trama

Os dados a transmitir têm por base uma estrutura de 32 *bytes*, constituída por três campos: - *número de sequência* cujo tamanho é de 2 *bytes* e representa o número do pacote a enviar; - *payload* cujo tamanho é de 29 *bytes* e corresponde ao campo dos dados; - *CRC* cujo tamanho é de 1 *byte*, e servirá como mecanismo de deteção de erros na transmissão.

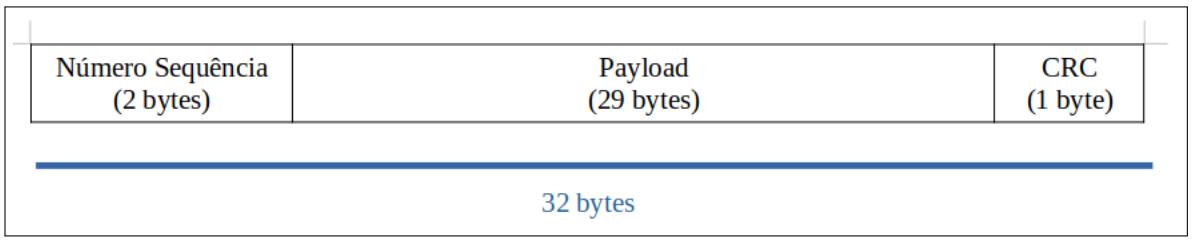


Figura 1: Estrutura da trama.

2.2. Protocolo de comunicação

Os protocolos de comunicação a utilizar serão os mesmos da fase anterior, ou seja, o protocolo SPI para a ligação ESP32-nRF24L01+ e a comunicação por RF para a ligação nRF24L01+-nRF24L01+.

Para a ligação **ESP32 - nRF24L01+**, a interface de comunicação implementada é o protocolo SPI, que é um protocolo de barramento de interfaces síncrono. Por oposição ao protocolo UART(utilizado na Fase 1), o protocolo SPI faz a transmissão de dados através de um sinal de *clock* (SCK - Serial ClocK) gerado pela placa ESP32, um sinal de envio de dados (MOSI - Master-Out/Slave-In), e para a receção de dados, a placa ESP32 continua a gerar ciclos do sinal SCK para que o módulo nRF24L01+ possa enviar informação pelo sinal de dados MISO (Master-In/Slave-Out).

Para a ligação **nRF24L01+ - nRF24L01+**, a interface de comunicação é a comunicação por radiofrequência, que consiste na emissão e receção de informação codificada e modulada num sinal eletromagnético. De forma, a implementar a comunicação por RF foi utilizada a *library* **RF24.h**, disponibilizada no *software* Arduino IDE.

Para que possa transmitir, o módulo nRF24L01+ foi desenhado para operar na banda de frequências ISM de 2,4 GHz e utiliza a modulação GFSK, que é uma extensão da modulação FSK(Frequency Shift-Keying). Abaixo encontram-se a figura que exemplifica a transmissão de dados com a modulação FSK e a figura que exemplifica a transmissão de dados com a modulação GFSK, respetivamente.

2.3. Controlo de Fluxo

De modo a garantir que não existe uma sobrecarga sobre o recetor, decidimos implementar o mecanismo *Stop-and-wait* em que após a transmissão de um pacote/trama, o emissor aguarda uma mensagem de confirmação da sua receção *ACK* antes de transmitir a trama seguinte, de modo a efetuar uma transferência fiável dos dados[1]. O tempo máximo que o emissor espera pela confirmação irá ser pré-definido, sendo que após esse intervalo de tempo a trama respetiva será retransmitida. De forma análoga, caso a trama recebida esteja corrompida, o recetor enviará uma mensagem *NACK* que informa o emissor que a trama chegou com erros de modo a ser retransmitida.

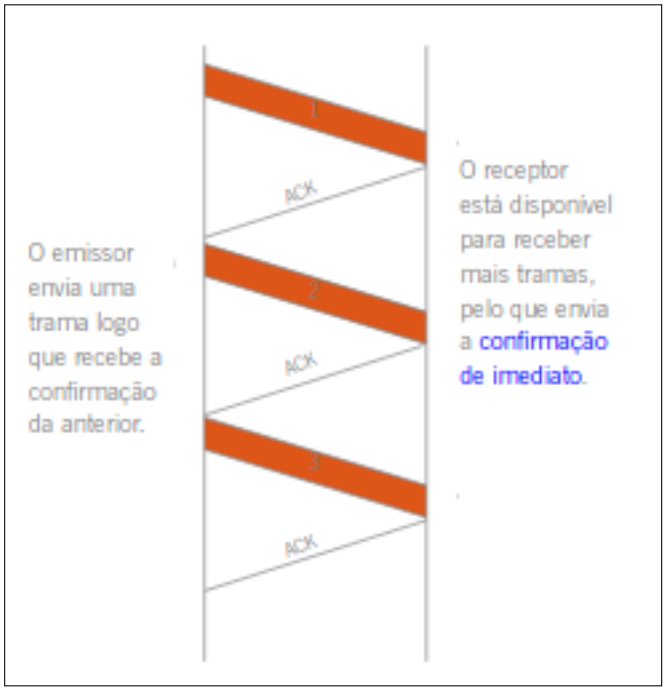


Figura 2: Protocolo Stop-and-wait.

Como podemos observar na imagem abaixo o Frame 3 não chegou ao recetor, ou seja, não teremos como enviar o Frame 4 sem a receção do ACK 3 e chegaríamos assim ao final do envio de dados, para que isto não aconteça utilizamos o *timeout*. O *timeout* permite que passado um tempo definido sem receber o ACK o pacote volte a ser reenviado seja por este não ter chegado ao recetor ou pelo ACK não ter chegado ao emissor.[2]

Isto levanta outro problema que pode ser a retransmissão infinita de dados caso, por exemplo, o recetor esteja desligado, para isto não acontecer poderemos definir um limite de retransmissões.

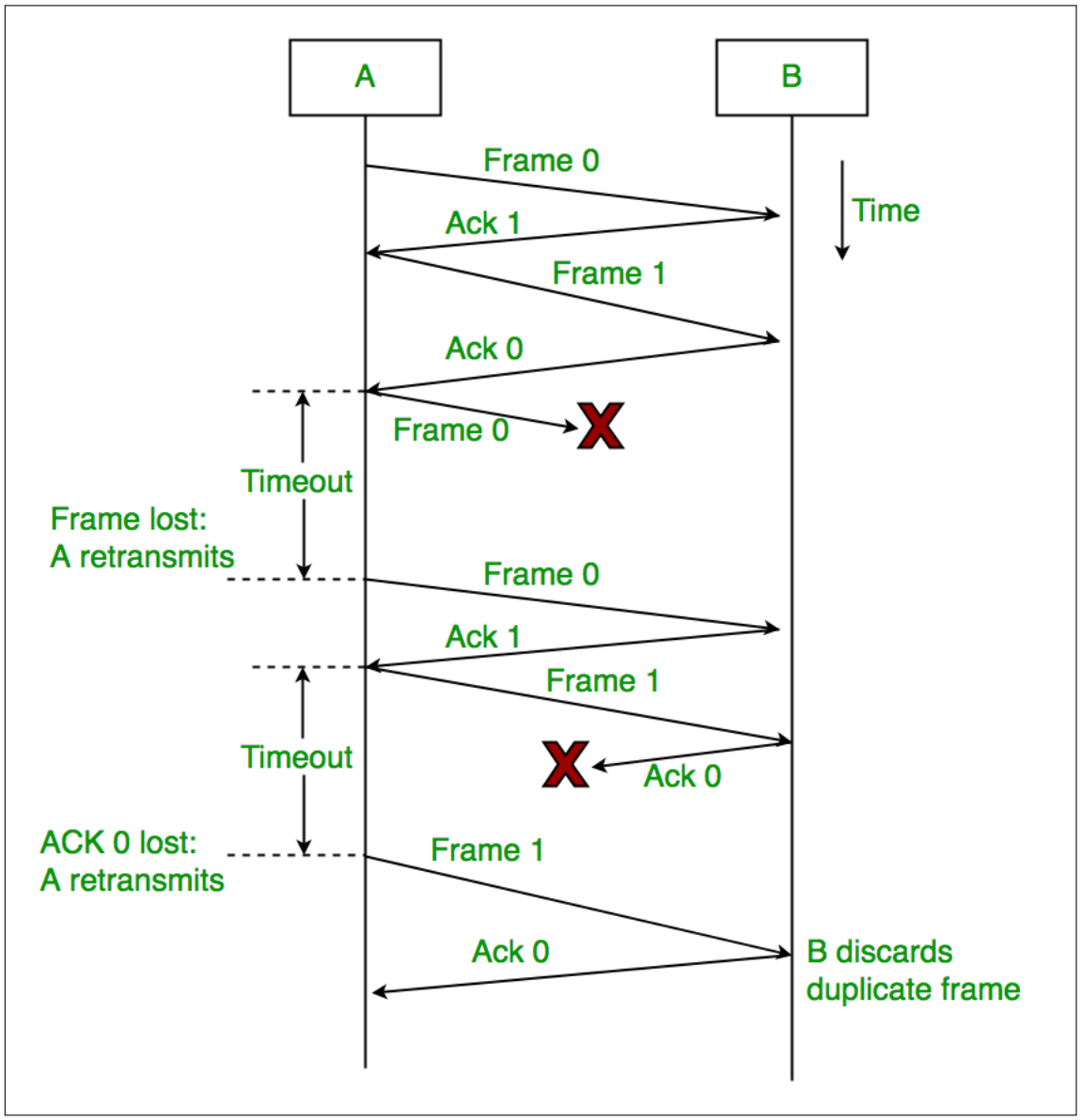


Figura 3: Protocolo Stop-and-wait com timeout.

3. Referências

[1] [online] Disponível em: `https://www.geeksforgeeks.org/stop-and-wait-arq/`

[Acedido a 5 de dezembro de 2020]

[2] [online] Disponível em: `https://en.wikipedia.org/wiki/Stop-and-wait_ARQ`

[Acedido a 5 de dezembro de 2020]