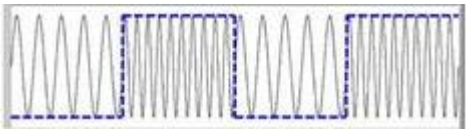


CAMADA FÍSICA DA COMPUTAÇÃO

ENGENHARIA DA COMPUTAÇÃO - Rodrigo Carareto – 0#07E4/02

PROJETO 4 – Checksum

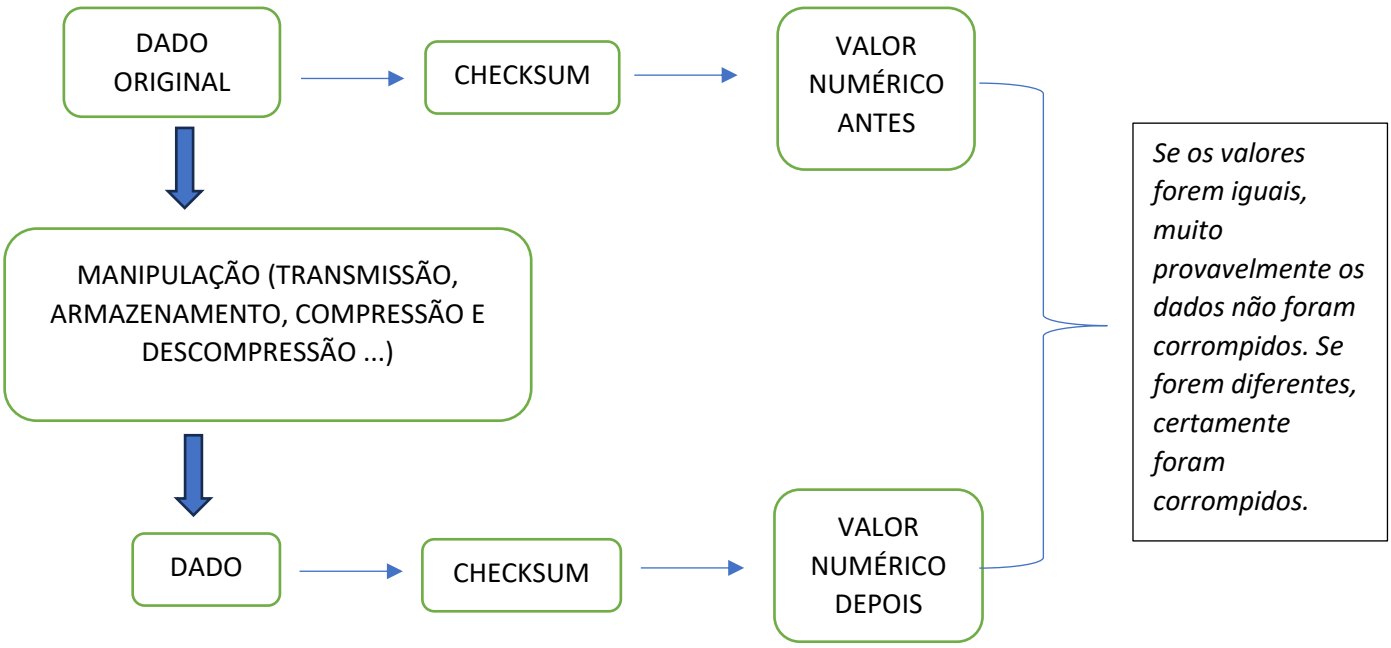
Nesse projeto iremos adicionar algumas funcionalidades no protocolo que você vem desenvolvendo. Uma delas é o checksum. Entende-se por Checksum (ou soma de verificação) um valor numérico gerado a partir de um conjunto de dados, usado para verificar a integridade desses dados. Ele é calculado por meio de uma função hash ou algoritmo matemático, criando um valor único baseado no conteúdo original. Se os dados forem alterados, o checksum resultante será diferente, indicando que houve uma modificação (intencional ou acidental). Para transmissão de dados, a ideia então é que a função hash seja aplicada aos dados originais, sendo o valor numérico gerado transmitido junto com os dados. No lado receptor, a função é aplicada novamente aos dados recebidos, e o valor numérico gerado pode então ser comparado ao gerado no lado emissor.

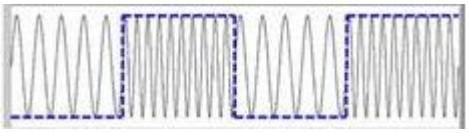
Como funciona:

1. Geração do checksum: Um algoritmo específico (como MD5, SHA-256, CRC32) processa os dados e gera um valor fixo.
2. Transmissão ou armazenamento: O dado e seu checksum são enviados ou armazenados juntos.
3. Verificação: O receptor recalcula o checksum a partir dos dados recebidos e compara com o original.
    - Se os valores coincidirem, os dados provavelmente estão intactos.
    - Se forem diferentes, houve alguma alteração ou erro.

Aplicações comuns:

- Verificação de integridade de arquivos (baixar um arquivo e conferir se não foi corrompido).
- Transmissão de dados (proteger contra erros em redes de comunicação).
- Segurança (detectar alterações não autorizadas em arquivos).





CAMADA FÍSICA DA COMPUTAÇÃO

ENGENHARIA DA COMPUTAÇÃO - Rodrigo Carareto – 0#07E4/02

Principais algoritmos

Existem vários algoritmos que são comumente utilizados como checksum, ou seja, associam um valor numérico a uma certa quantidade de dados. Aqui estão alguns exemplos:

Mais comuns por categoria

Algoritmo	*Bits (do valor gerado)*	*Aplicação Principal*
CRC32	32	Redes, compressão de arquivos (ZIP, RAR)
MD5	128	Verificação de arquivos (antigo, inseguro para segurança)
SHA-1	160	Git, certificados digitais (inseguro para criptografia)
SHA-256	256	Segurança, Blockchain, SSL/TLS
BLAKE2	256+	Alternativa rápida ao SHA-2
Adler-32	32	Checksum simples (mais rápido que CRC32, menos seguro)

Enunciado do projeto

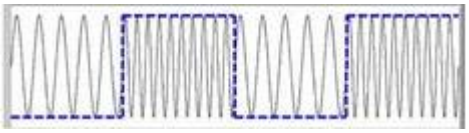
Você deverá implementar um mecanismo de verificação de integridade da transmissão (checksum). Isso deve ser feito através de um CRC-16. Em aula, seu professor explicou como funciona esse algoritmo, e você não precisará implementá-lo, apenas usar alguma biblioteca que o contenha. Caso queira relembrar como funciona esse algoritmo, voce pode pesquisar em algum chat com AI.

1. Você deverá transmitir, em algum pedaço de seu header, 2 bytes contendo o CRC-16 do respectivo **payload** (apenas o payload) sendo transmitido. Todos os pacotes conterão o CRC do payload.
2. Caso o lado que recebe o pacote (server) detecte uma incoerência entre o CRC enviado e o CRC calculado após o recebimento, este deve retornar uma mensagem (pacote) informando o erro de CRC, solicitando o reenvio do pacote. Se tudo estiver ok, a resposta deve ser informando que não houve incoerências e a transmissão seguir.
3. Além do CRC, tanto do lado cliente como o lado server, devem gerar um arquivo txt com um log da transmissão. Tanto do lado server, como do client, uma linha deve ser criada no arquivo **toda vez que um pacote é enviado ou recebida**.

Cada linha deve conter:

- Instante do envio ou recebimento
- É envio ou recebimento
- Tipo de mensagem (de acordo com o protocolo: dados? Ok? Erro?)
- Tamanho de bytes total
- Pacote enviado (se for pacote do tipo dados)
- Total de pacotes (se for pacote do tipo dados)
- CRC do payload para mensagem (se for pacote do tipo dados)

Você pode formatar seu arquivo como quiser. As linhas abaixo são um **exemplo** feito por um aluno de um arquivo gerado pelo client. Nesse exemplo, o tipo de pacote dados é representado pelo número 3. O pacote do tipo resposta ok é representado pelo número 4. Os pacotes tinham até 128 bytes.



CAMADA FÍSICA DA COMPUTAÇÃO

ENGENHARIA DA COMPUTAÇÃO - Rodrigo Carareto – 0#07E4/02

29/09/2020 13:34:23.089 / envio / 3 / 128 / 1 / 23/ F23F
29/09/2020 13:34:23.230 / receb / 4 / 14
29/09/2020 13:34:23.569 / envio / 3 / 128/ 2 / 23/ FE3A
29/09/2020 13:34:23.885 / receb / 4 / 14
29/09/2020 13:34:24.029 / envio / 3 / 128 / 3 / 23/1802
29/09/2020 13:34:24.230 / receb / 4 / 14

Quando o envio de um pacote ocorrer com algum tipo de erro, seja a ordem errada na transmissão dos pacotes (por exemplo, pular um pacote no envio), seja por CRC, esse problema deve ser corrigido e a transmissão terminar com sucesso!

Voce deverá implementar erros propositais (pode ser hard coded) para gerar arquivos de log em diversas condições:

- Transmissão sem nenhuma intercorrência.
- Transmissão com erro na ordem dos pacotes enviados pelo client.
- Transmissão com erro de CRC.
- Transmissão com interrupção física de envio e reinício (fios retirados e reconectados) !

Seu professor irá solicitar transmissões com esses problemas e verificar os logs do server e do cliente!

Durante a apresentação

Ao solicitar a correção, seu professor poderá fazer perguntas e testar o funcionamento do protocolo perante algumas intercorrências, como desligamento e religamento de jumpers. Seu código, se construído de maneira correta, será robusto a tais intercorrências. Além disso, os arquivos logs serão analisados para transmissões com os erros mencionados anteriormente e os arquivos de log verificados!

As notas de C a A+ serão dadas de acordo com a robustez, precisão, funcionalidades e velocidade das transmissões

A entrega deve ser feita até 18/09/2025