Protocolos de enlace

Comunicação de Dados

Diego Fumagalli - 161150923
Jian Furquim - 161152063
Henrique Fan - 161152061

Universidade Federal do Pampa - UNIPAMPA

Agenda

- Introdução
- Metodologia
- Resultados
- Conclusão

Introdução

Esse projeto é uma implementação de um algoritmo em Python para a avaliação de desempenho de envio e recepção de informação. Buscando fornecer dados e estimativas de tempo de envio, recepção e erros, dependendo do tamanho do frame e tamanho da mensagem.

Introdução

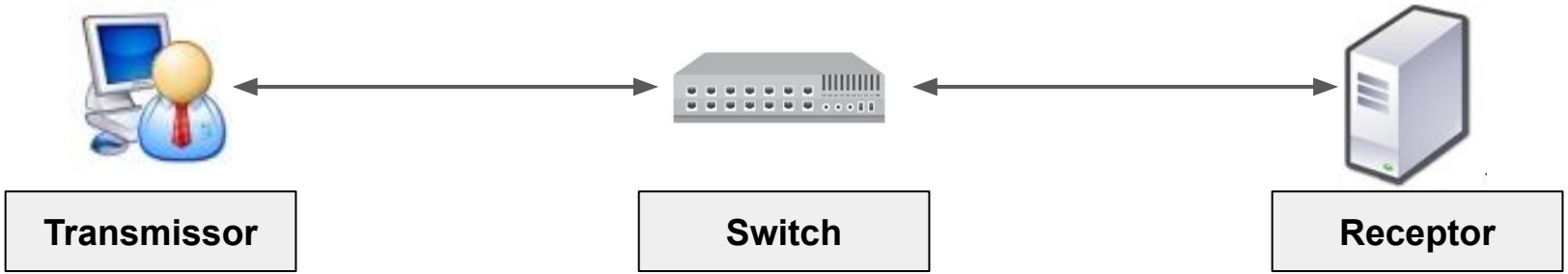
- Implementação de protocolos da camada enlace:
  - Enquadramento;
  - Controle de erro de modificação;
  - Controle de erro de perdas.
- Emulado através do protocolo de nível de transporte UDP.

Introdução

Metodologia

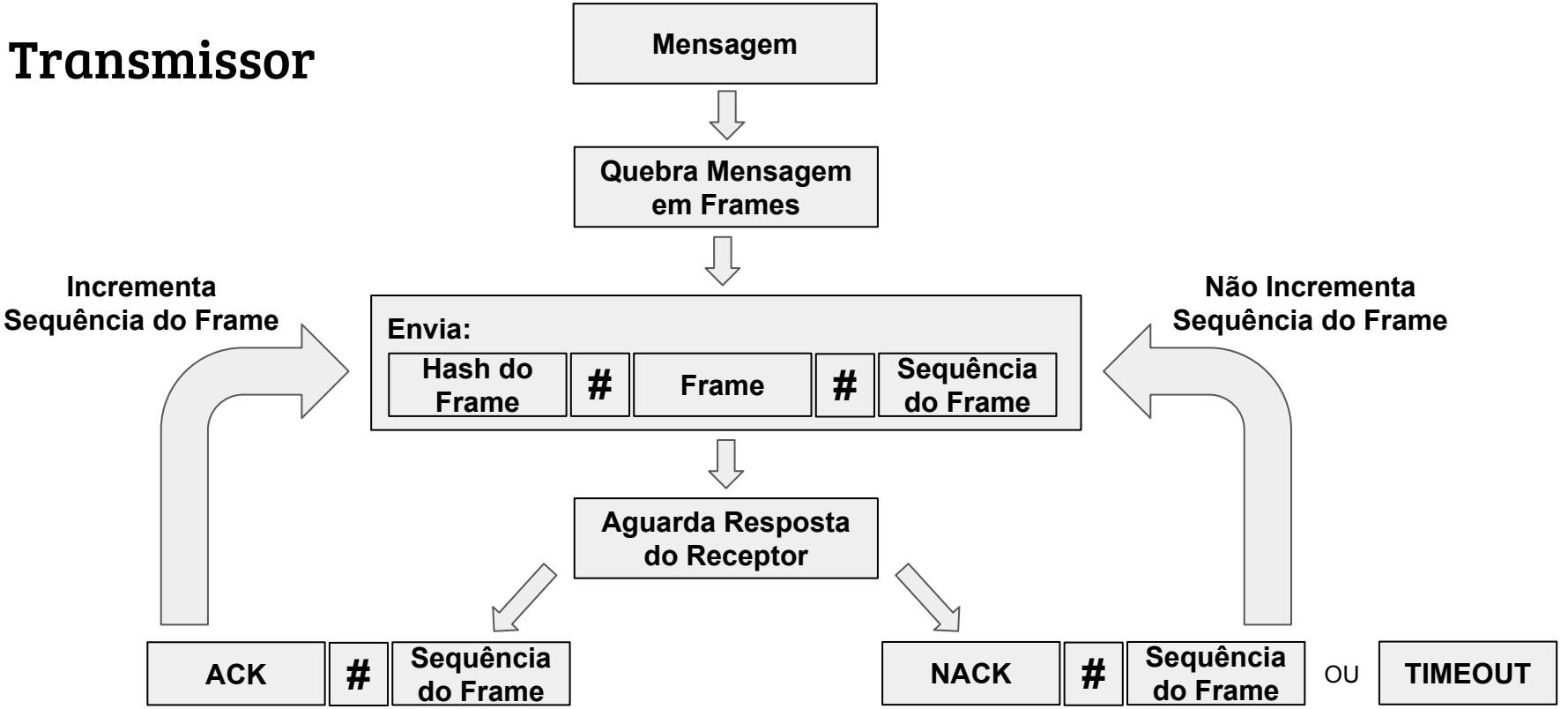
Cenário

A comunicação entre o transmissor e o receptor se dá a partir de um topologia em linha.



Metodologia

Transmissor



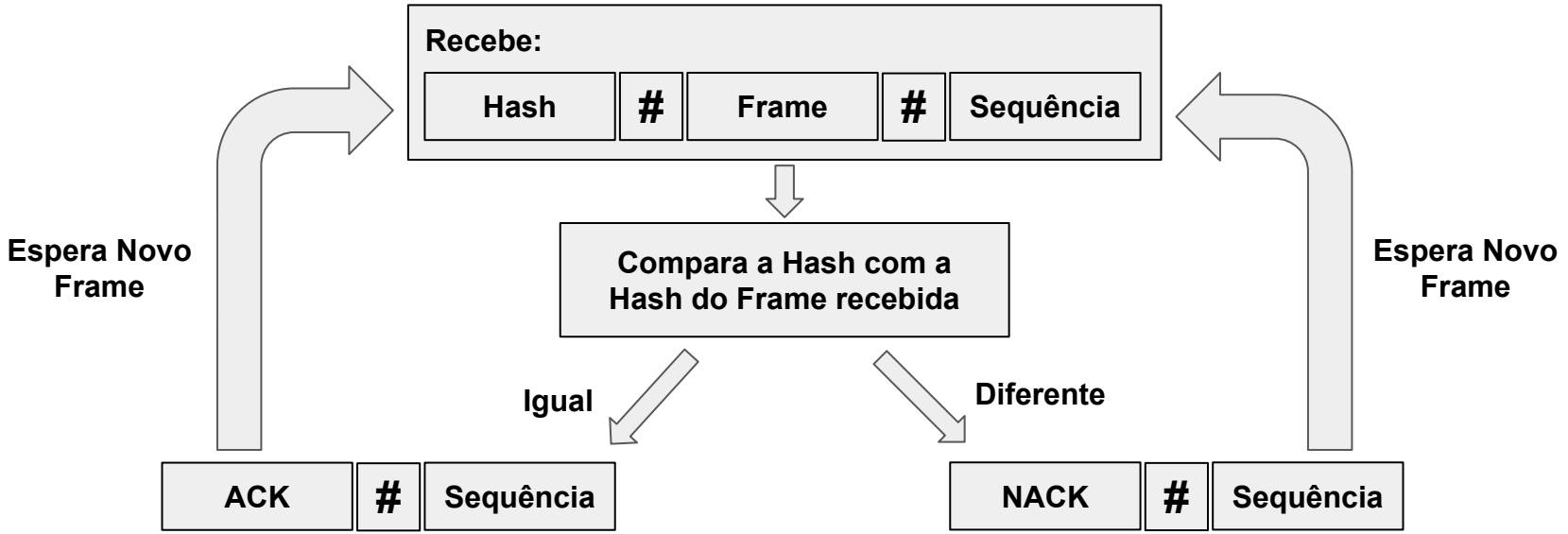
Metodologia

```
time_begin = datetime.datetime.now()
while i < len(data):

    frame = frameCurrent(i, data, args.frame)

    h = hashlib.md5(bytes(frame, ENCODE)).hexdigest()

    if random.random() < args.loss: # porcentagem de erro
        s.settimeout(None)

        logging.info('{} --- {:05d}'.format(error, countFrame, end=''))

        msg = h + '#' + error + '#' + '%05d'%countFrame
        s.send(bytes(msg, ENCODE))
    else:
        s.settimeout(None)

        logging.info('{} --- {:05d}'.format(frame, countFrame, end=''))

        msg = h + '#' + frame + '#' + '%05d'%countFrame
        s.send(bytes(msg, ENCODE))

    s.settimeout(1)
    try:
        recv = s.recv(1024).decode(ENCODE)

        response, seq = recv.split('#')

        logging.info('{} {}'.format(response, seq))

        if response == 'nack':
            continue

        i += args.frame
        countFrame = int(seq) + 1

    except Exception as e:
        logging.info('exception: {}'.format(e))
        continue

s.send(b'')
time_end = datetime.datetime.now()
```

- Estância o tempo
- Separa a mensagem enviada em frames;
- Hash dos frames;
- Envia os frames com a Hash e a sequência;
- Possibilidade de erro;
- Timeout 1 s para reenviar o frame;
- ACK, manda próximo frame;
- NACK, reenvia frame.

Metodologia

Receptor



Metodologia

```
data = ''
while True:

    recv  = s.recv(1024).decode(ENCODE)

    if not recv:
        break

    h, frame, seq = recv.split('#')

    if random.random() < args.fault: # chance de não responder
        continue
    else:

        hFrame = hashlib.md5(str(frame).encode(ENCODE)).hexdigest()

        if h == hFrame:
            logging.info('{} == {} {} {}'.format(h, hFrame, frame , seq))

            data += frame
            s.send(bytes('ack#'+str(seq), ENCODE))
        else:
            logging.info('{} != {} {} {}'.format(h, hFrame, frame , seq))
            s.send(bytes('nack#'+str(seq), ENCODE))


logging.info('Enviado: {}'.format(data))

s.send(bytes(data, ENCODE))
```

- Recebe os frames com a Hash e a sequência;
- Se for vazio, sai do loop e envia mensagem vazia;
- Mensagem não vazia, verifica hash e manda confirmação para transmissor;
- Monta toda a mensagem e envia para o transmissor.

Metodologia

Ambiente

Para a execução dos teste foi usado um máquina com as seguintes configurações:

- Intel® Core™ i5-5200U CPU - 2.20GHz × 4;
- 8 GB de memória RAM;
- Ubuntu 20.04.1 LTS - (64-bits);
- Vagrant - 2.2.10;
- VirtualBox - v6.1.10_Ubuntu.

- Máquina virtual:
  - 2 Processadores;
  - 2 GB de memória RAM;
  - Ubuntu 18.04 - (64-bits).
  - Mininet - 2.3.0d6

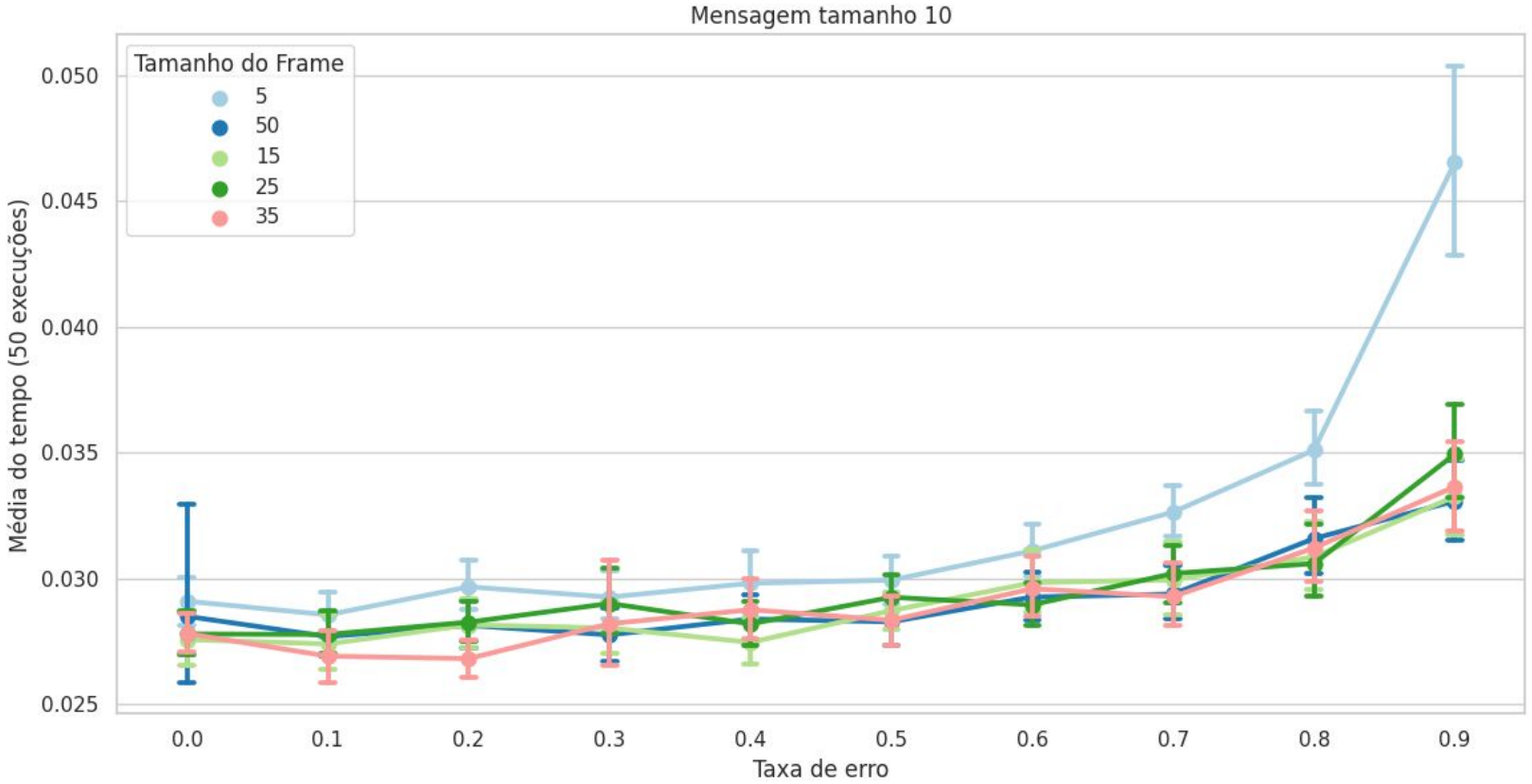
Metodologia

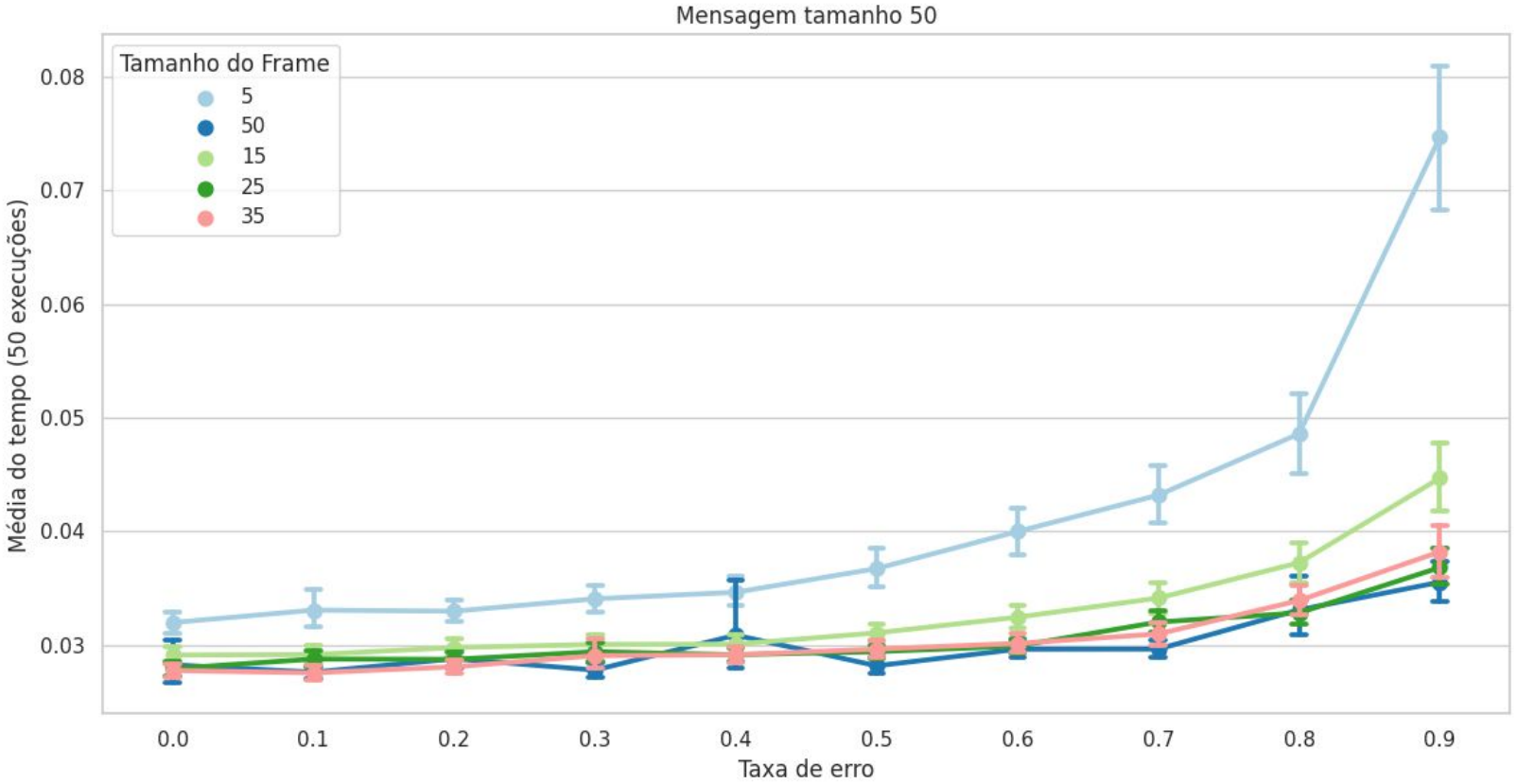
Parâmetros		Descrição	Variáveis no código	Padrão
--count	-c	Quantidade de execuções	DEFAULT_COUNT	1
--mbps	-b	Taxa de transmissão	DEFAULT_BW_MBPS	1
--switches	-s	Quantidade de switches	DEFAULT_SWITCHES	1
--delays	-d	Tempo de delay	DEFAULT_DELAY_MS	0
--listmessages	-lm	Lista dos tamanhos da menssagens	DEFAULT_SIZE_MESSAGE	[10]
--listframes	-lf	Lista dos tamanhos dos frames	DEFAULT_SIZE_FRAME	[2]
--fault	-pl	Probabilidade de falha (chance de receptor não responder)	DEFAULT_PF	0.0
--loss	-pl	Probabilidade de perda (chance de perda da mensagem)	DEFAULT_PL	0.0

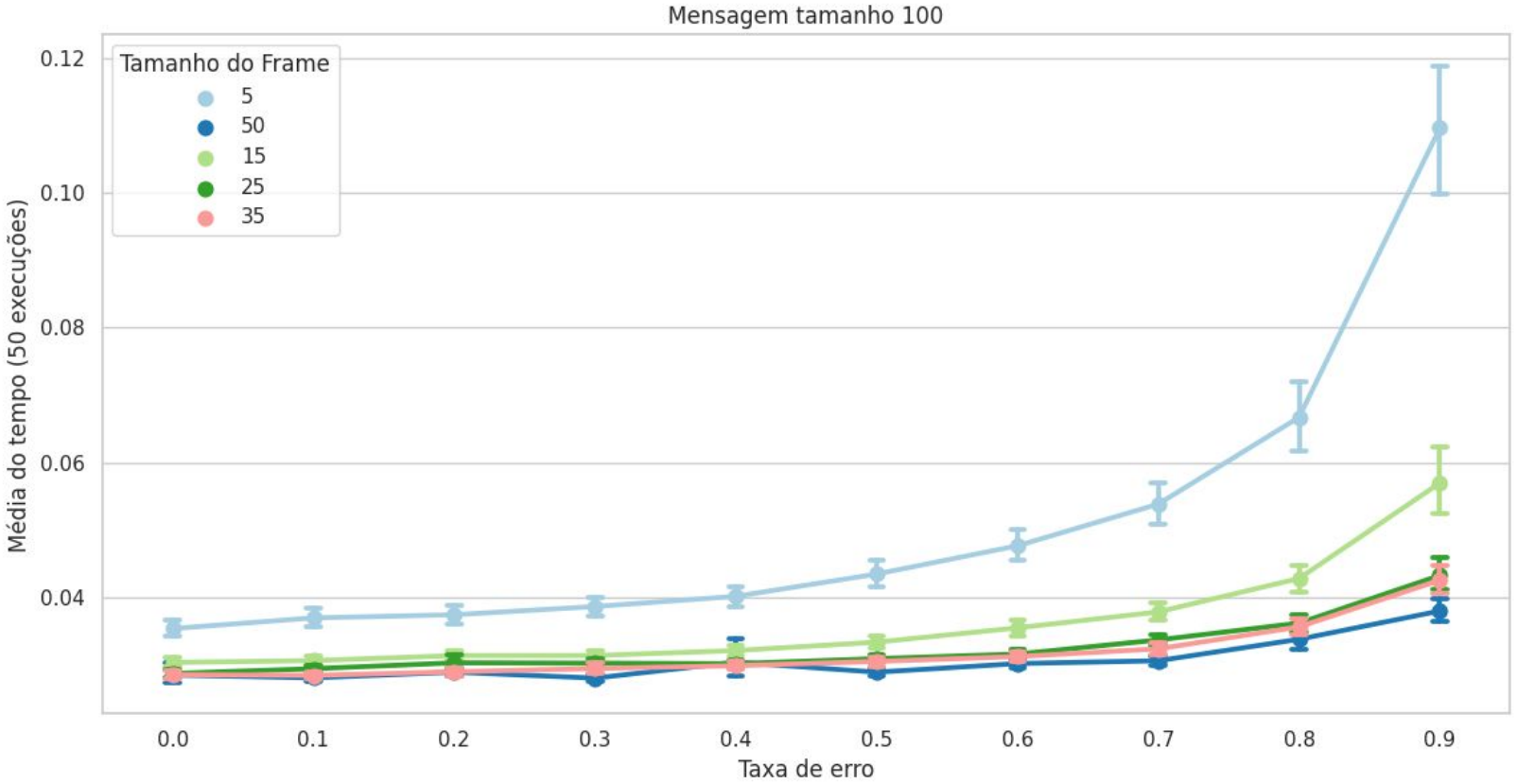
Resultados

- Comando para usado no experimento:

$ sudo python3 lab.py -lm 10 50 100 -lf 5 15 25 35 50 -pl 0.1 -c 50


Mensagem tamanho 10
Tamanho do Frame
5
50
15
25
35
Média do tempo (50 execuções)
0.050
0.045
0.040
0.035
0.030
0.025
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Taxa de erro


Mensagem tamanho 50
Tamanho do Frame
5
50
15
25
35
Média do tempo (50 execuções)
0.08
0.07
0.06
0.05
0.04
0.03
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Taxa de erro


Mensagem tamanho 100
Tamanho do Frame
5
50
15
25
35
Média do tempo (50 execuções)
0.12
0.10
0.08
0.06
0.04
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Taxa de erro

Conclusão

- Percebe-se que, a maior influência no tempo total da transmissão da mensagem é a quantidade de frames em que ela foi dividida.
- Quanto maior a mensagem maior o tempo de execução, em contraste, quanto menor o frame maior o tempo de execução. Pois um está relacionado ao outro
- Como sugestão futura, poderá ser implementado controle de fluxo de dados.
- Como melhoria do trabalho, sugere-se a automatização de geração de gráficos.

Obrigado !