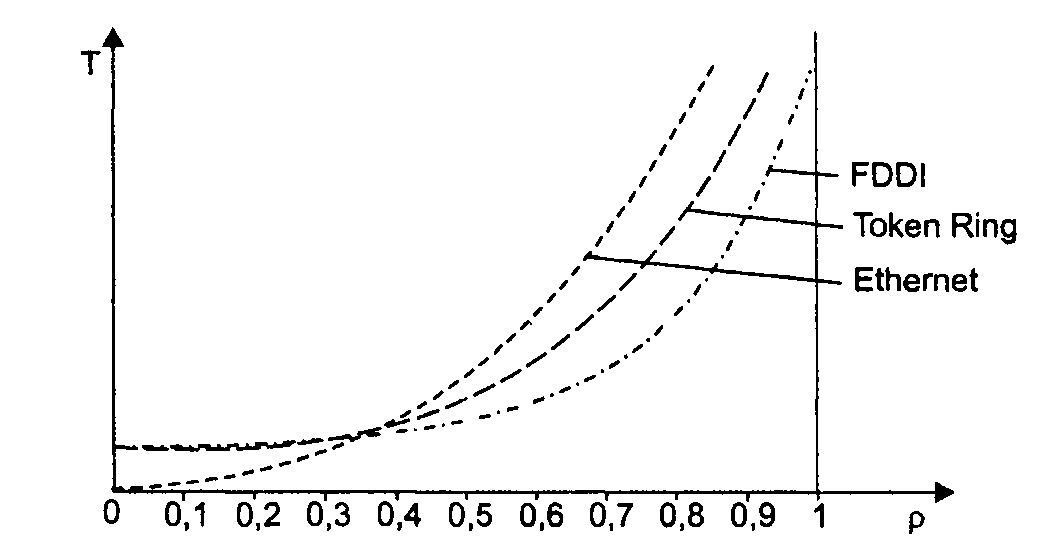
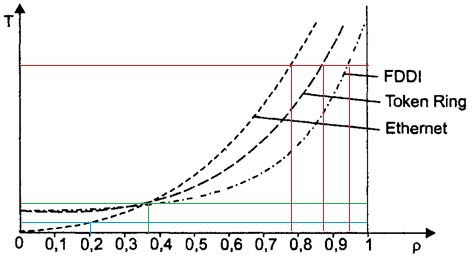
ЛАБОРАТОРНАЯ РАБОТА № 4

Передача дискретных данных по линиям связи

Задание.

1. Построить экспериментальный график зависимости среднего времени ожидания доступа к среде для протокола Ethernet (+ «парочки других») от коэффициента использования сети. Определить параметры, которые влияют на вид этой зависимости.  
   Найти в интернете некий график и при сдаче отчёта мочь прочитать его.





1. Рассчитать эффективную и реальную пропускную способность сети Ethernet 100BaseT при минимальной и максимальной длине кадра.  
     
   Кадр мин длины = 8 + 14 + 46 + 4 байт = 72 байта (576 бит)  
   Кадр макс длины = 8 + 14 + 1500 + 4 байт = 1526 байт (12208 бит)  
     
   Скорость 100BaseT = 100Мбит/с = 100 000 000 бит/с = 100 000 бит/мс = 100 бит/мкс  
   Время передачи кадра мин длины = 5.76 мкс  
   Время передачи кадра макс длины = 122.08 мкс  
   Технологическая пауза = 9.6 мкс  
     
   Время для мин длины = 15.36 мкс  
   Время для макс длины = 131.68 мкс

Частота мин кадров = 1 / 0.00001536 = 65 035.5 кадров/с

Частота макс кадров = 1 / 0.00013168 = 7591.5 кадров/с

Эффективная пропускная способность = полезная нагрухка х частота х 8 бит

**ЭПС мин длины = 46 х 65 035.5 х 8 = 23 933 064‬ бит/с = 23.93 Мбит/с**

**ЭПС макс длины = 1500 х 7591.5 х 8 = 91 098 000 бит/с = 91.1 Мбит/с**

Пусть реальная пропускная способность имеет коэффициент 0.8 в отношении к эффективной

**РПС мин длины = 23.93 х 0.8 = 19.14 Мбит/с  
 РПС макс длины = 91.1 х 0.8 = 72.88 Мбит/с**

1. Ознакомиться с принципами работы протокола ICMP. Создать приложение, реализующее функции, аналогичные утилите ping с использованием библиотеки icmp.dll

from icmplib import ping, exceptions

import argparse

import signal

import sys

import time

def main():

    parser = argparse.ArgumentParser(

        description="Эмуляция команды ping",

        add\_help=True

    )

    parser.add\_argument('destination', help='Адрес узла для проверки')

    parser.add\_argument('-t', action='store\_true', help='Непрерывный пинг (до Ctrl+C)')

    parser.add\_argument('-n', type=int, help='Число запросов (по умолчанию: 4)')

    parser.add\_argument('-w', type=int, help='Таймаут ответа (мс)')

    parser.add\_argument('-S', type=str, help='Адрес источника')

    args = parser.parse\_args()

    pings = args.n or (None if args.t else 4)

    ping\_kwargs = {

        'address': args.destination,

        'interval': 1,

        'timeout': (args.w / 1000) if args.w else 0,

        'source': args.S,

        'privileged': True

    }

    print(f"\nОбмен пакетами с {args.destination}:")

    seq = 1

    total\_sent = 0

    total\_received = 0

    times = []

    def print\_stats():

        print(f"\nСтатистика ping:")

        lost = total\_sent - total\_received

        loss\_percent = (lost / total\_sent) \* 100 if total\_sent > 0 else 0

        print(f"    Отправлено = {total\_sent}, Получено = {total\_received}, Потеряно = {lost} ({loss\_percent:.1f}% потерь)")

        if times:

            print(f"    Время (мс): мин. = {min(times):.0f}, макс. = {max(times):.0f}, сред. = {sum(times)/len(times):.0f}")

    def signal\_handler(signum, frame):

        print\_stats()

        sys.exit(0)

    # Ctrl+C

    signal.signal(signal.SIGINT, signal\_handler)

    signal.signal(signal.SIGBREAK, signal\_handler)

    try:

        while True:

            for i in range(pings):

                result = ping(\*\*ping\_kwargs, id=seq)

                if i:

                    time.sleep(ping\_kwargs['interval'])

                total\_sent += 1

                if result.packets\_received:

                    total\_received += 1

                    times.append(result.avg\_rtt)

                    print(f"Ответ от {result.address}: время={result.avg\_rtt:.0f}мс")

                else:

                    print("Превышен таймаут ожидания.")

            if not args.t:

                break

        print\_stats()

        sys.exit(0)

    except KeyboardInterrupt:

        signal\_handler(None, None)

    except exceptions.NameLookupError:

        print(f"Ошибка: Не удалось найти имя узла \"{args.destination}\"")

if \_\_name\_\_ == '\_\_main\_\_':

    main()