**Теоретические вопросы**

1. Протоколы и веб-сервисы в распределенных системах.
2. Групповые взаимодействия и рассылка. Непрямое взаимодействие в распределенных системах.
3. Обнаружение отказов. Масштабирование.
4. Параллельная обработка данных в распределенных системах

**Практическое задание**

1. Реализовать запросы к удаленному серверу по протоколу HTTP с использованием утилит **telnet**, **curl** к следующим ресурсам:

<http://www.mgpu.ru/>

<https://bmstu.ru/>

<https://cbr.ru/>

telnet –

curl –

get / HTTP/1.1/

Host: hse.ru

Материалы для подготовки размещены на ресурсе <https://github.com/BosenkoTM/Distributed_systems/tree/main/practice/2024/lw_02>

2. Реализовать рассылку сообщений с помощью IP Multicast. На примере файлов [socket\_multicast\_sender.py](https://github.com/BosenkoTM/ds_practice/blob/main/exercises/winter_semester_2021-2022/02-multicast/multicast/socket_multicast_sender.py) и [socket\_multicast\_receiver.py](https://github.com/BosenkoTM/ds_practice/blob/main/exercises/winter_semester_2021-2022/02-multicast/multicast/socket_multicast_receiver.py) реализовать рассылку сообщений.

import socket

import struct

import time

# Настройки Multicast

MCAST\_GRP = "224.1.1.1"  # Multicast адрес

MCAST\_PORT = 5007        # Порт для отправки сообщений

TTL = 1                  # Время жизни пакета (1 означает только в локальной сети)

# Создаем сокет для отправки данных через UDP

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM, socket.IPPROTO\_UDP)

# Привязываем сокет к любому локальному адресу и указанному порту

sock.bind(('', MCAST\_PORT))

# Настраиваем подписку на Multicast группу

mreq = struct.pack("4sl", socket.inet\_aton(MCAST\_GRP), socket.INADDR\_ANY)

sock.setsockopt(socket.IPPROTO\_IP, socket.IP\_ADD\_MEMBERSHIP, mreq)

print("Ожидание сообщений от Multicast группы...")

try:

    while True:

        # Получаем сообщение

        data, addr = sock.recvfrom(1024)  # Размер буфера 1024 байта

        print(f"Сообщение от {addr}: {data.decode()}")

except KeyboardInterrupt:

    print("\nПрием сообщений остановлен.")

    sock.close()

# Настраиваем TTL (Time-To-Live) пакета

ttl = struct.pack('b', TTL)

sock.setsockopt(socket.IPPROTO\_IP, socket.IP\_MULTICAST\_TTL, ttl)

message = input() #"Привет, Multicast группа!"

# Отправляем сообщение в Multicast группу

sock.sendto(message.encode(), (MCAST\_GRP, MCAST\_PORT))

print(f"Отправлено сообщение: {message}")

time.sleep(2)  # Отправляем сообщение каждые 2 секунды

Материалы для подготовки размещены на ресурсе

<https://github.com/BosenkoTM/Distributed_systems/tree/main/d-zadanie/04-multicast>

3. Реализовать механизмы безопасности в распределенной системе с использованием сертификатов X.509 для аутентификации. Создайте распределенную систему, состоящую из:

* Сервера распределенной системы.
* координатора распределенной системы.
* одного клиента распределенной системы HTTPS.
* одного клиента распределенной системы HTTP.

Материалы для подготовки размещены на ресурсе <https://github.com/BosenkoTM/Distributed_systems/tree/main/practice/2024/lw_05>

4. Определить сходимость **SWIM,** интервалы конвергенции (указать процент конвергенции, при котором график имеет наибольшее отклонение от гладкой кривой) компьютерной распределенной сети при следующих входных параметрах:

**GOSSIP FANOUT** -10, 5, 3 nodes.

**GOSSIP INTERVAL** – 0.1

**NODES** = 50, 100

**PACKET LOSS** – 10%, 50%.

Материалы для подготовки размещены на ресурсе

<http://95.131.149.21/moodle/mod/assign/view.php?id=935>