Задание «Протокол DNS»

Загружать решения в виде Zip архива в проект Smart LMS под названием HW3.
В архиве должна быть папка с проектом Java (см. ниже).
Название ZIP архива должно совпадать с Вашей фамилией.

Мотивация. Для углубленного изучения протокола DNS необходима практика в виде разработки небольшой программной утилиты для выполнения задач, связанных с протоколом DNS.

Нужно использовать сетевую библиотеку PCAP (версия для Windows – NPCAP). Это чрезвычайно популярная библиотека. Например, утилита nmap (фигурирует в нескольких известных художественных фильмах), использует сетевую библиотеку PCAP:
https://nmap.org/movies/
Ее также используют много современных больших проектов:
https://en.wikipedia.org/wiki/Pcap

Оценивание задания. Во-первых, код будет анализироваться на точность выполнения требований этого задания. Затем будет учитываться корректность и качество кода (форматирование, обработка исключений, иные стандартные элементы качества кода). Дополнительно может быть проведена устная защита выборки домашних работ. Наконец, Ваша оценка будет дополнена тестом (будет проходить на одном из семинаров по материалам лекций и семинаров).
Вклад данного задания в оценку – до 80%, вклад теста – до 20%.

Содержание задания.

Задание по содержанию похоже на предыдущее задание с протоколом ARP и отличается составом выполняемых приложением функций и упаковкой DNS пакетов.

Реализовать приложение на Java с использованием библиотеки PCAP4J. Управление приложением должно происходить путем ввода пользователем команд с консоли (как именно – подумайте и спроектируйте самостоятельно). Следует выполнять следующие задачи:

- Захват всех пакетов DNS и вывод их на консоль (включите «неразборчивый (*PROMISCUOUS*)» режим Вашего сетевого интерфейса) – интерпретируйте формат и содержимое захваченных кадров (в лекции по протоколу DNS, см. OneDrive, содержится описание заголовка и протокола DNS. Используйте эту информацию для интерпретации захваченных кадров).
- Пользователь задает доменное имя *D*. Вашему приложению необходимо найти IP-адрес почтового сервиса *IPmx* (ресурсная запись *MX*, поиск осуществляется в два шага), с помощью которого нужно отправлять почту. Вывести на консоль результат в виде *D -> IPmx* (лишнюю информацию не выводить). Если вариантов несколько, вывести каждый из них на отдельной строке.
- Возьмите на сайте **http://www.root-servers.org/** адрес любого корневого сервера. Обратитесь с помощью Вашего приложения к нему за поиском IP адресов для доменов cnn.com, hse.ru, draw.io. Что выдает в своем ответе корневой сервер? Проанализируйте. Обратитесь к своему Интернет-провайдеру за той же информацией. Какой ответ Вы получили? Проанализируйте.

Внимание:

- Программа при старте должна выводить на консоль перечень команд, которые она поддерживает.
- Каждое под-задание данного задания должно реализовываться в отдельном классе/методе.
- Должно быть четко видно, в каком месте исходного кода реализована та либо иная функция**.**
- **Можно создать текстовый файл, который содержит перечень функций и указание на место кода, где функция реализована.**

Лекции и задания находятся на OneDrive.

Ход работы и фрагменты кода в помощь.

- Пожалуйста, создавайте maven/gradle проект.
- Установите WireShark, в который входят драйвера для WinPCAP.
- Запустите WireShark либо ipconfig -all и найдите
    - IP адрес и MAC адрес своего сетевого интерфейса, по которому Ваше устройство подключено к Интернет
    - IP адрес нужно будет указать в InetAddress.*getByName*("192.168.1.6"); вместо "192.168.1.6"
    - MAC адрес нужно будет указать в MacAddress.getByName("fe:00:01:02:03:04"); вместо “ fe:00:01:02:03:04”
- Ваши IP и MAC можно hardcode

Язык для выполнения работы – Java; нужно использовать только PCAP (low-level API).
Использовать Oracle JDK 21.
Код на другом языке либо без низкоуровневого PCAP не принимается.

Зависимости для Maven проекта:

```
<dependency>
    <groupId>org.pcap4j</groupId>
    <artifactId>pcap4j-core</artifactId>
    <version>[1.0, 2.0)</version>
</dependency>
<dependency>
    <groupId>org.pcap4j</groupId>
    <artifactId>pcap4j-packetfactory-static</artifactId>
    <version>[1.0, 2.0)</version>
</dependency>
```

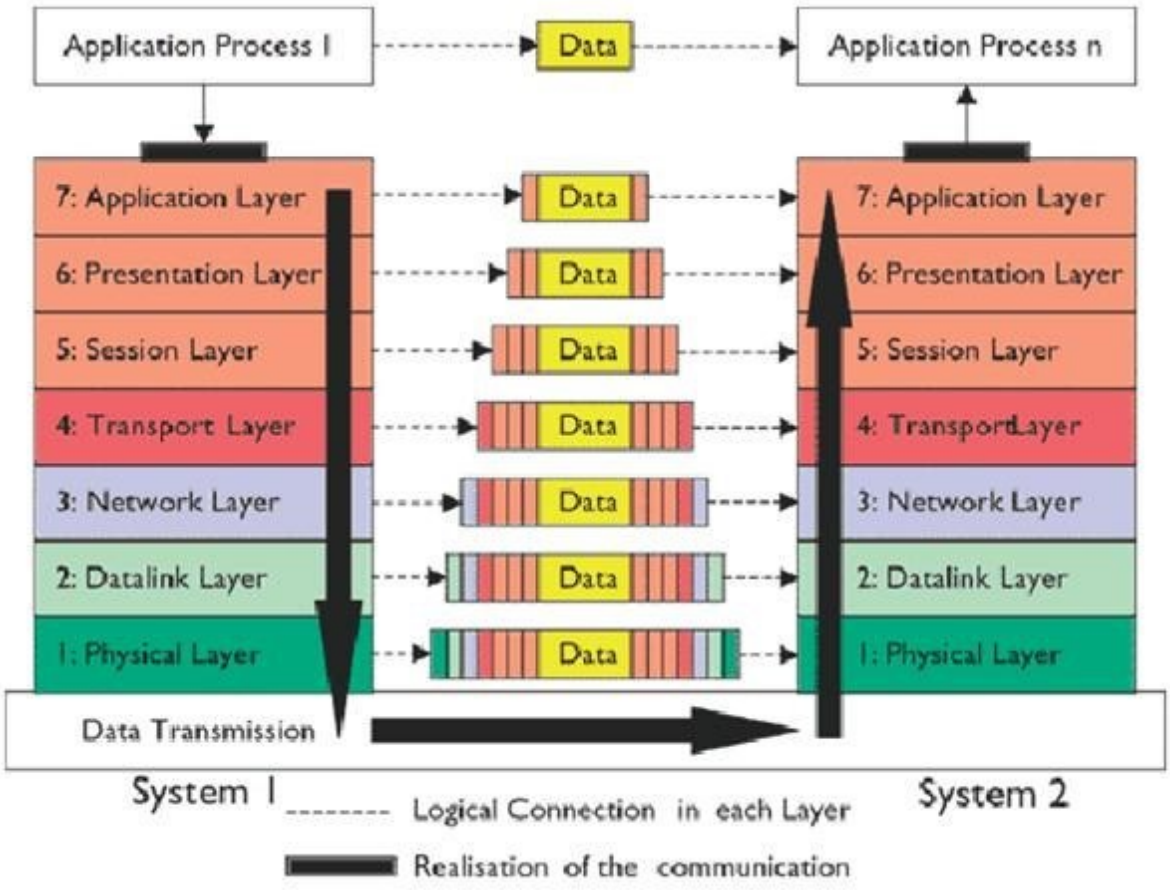
Минимальный код на Java для работы с WinPCAP (ведется захват всех сообщений по протоколу ARP):

```
package networks2024;

import org.pcap4j.core.*;
import org.pcap4j.packet.ArpPacket;
import org.pcap4j.packet.Packet;

import java.io.EOFException;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.concurrent.TimeoutException;


public class PcapARP {
    public static void main(String args[]) throws UnknownHostException,
PcapNativeException, EOFException, TimeoutException, NotOpenException {
        InetAddress addr = InetAddress.getByName("192.168.1.6");
        PcapNetworkInterface nif = Pcaps.getDevByAddress(addr);

        int snapLen = 65536;
        PcapNetworkInterface.PromiscuousMode mode =
PcapNetworkInterface.PromiscuousMode.PROMISCUOUS;
        int timeout = 10000;
        PcapHandle handle = nif.openLive(snapLen, mode, timeout);

        while(true) {
            Packet packet = handle.getNextPacketEx();

            ArpPacket arpPacket = packet.get(ArpPacket.class);
            if(arpPacket !=null) {
                System.out.println(arpPacket);
            }
        }
    }
}
```

Пример отправки ARP пакетов (PCap4J):

https://github.com/kaitoy/pcap4j/blob/v1/pcap4j-sample/src/main/java/org/pcap4j/sample/SendArpRequest.java

Для того, чтобы отправить DNS пакет, нам нужно создать все пакеты нижележащих уровней, том числе и Ethernet фрейм (см. рисунок ниже).



Builder pattern (fluent pattern):

```
public class BuilderUsage {
    public void use() {
        AClass.newBuilder()
                .setWeight(20)
                .defaultSpeed()
                .build();
    }
}

public static class Builder implements
        AClassDefinition,
        AClassSpeedDefinitionStage,
        AClassFinalStage {
    private int weight;
    private int speed;

    public AClassSpeedDefinitionStage setWeight(int weight) {
        this.weight = weight;
        return this;
    }

    public AClassFinalStage setSpeed(int speed) {
        this.speed = speed;
        return this;
    }

    public AClassFinalStage defaultSpeed() {
        this.speed = 10;
        return this;
    }

    public AClass build() {
        return new AClass(weight, speed);
    }
}
```