Java Project “redes”

Redes means “network” in portuguese.

Java,
Visual Studio Code

Subject:
Network between computers and routers.

> Create the respective tests, JavaDocs and execution classes.

> You must create a class with the respective main method for each problem and leave it in your project folder.

Notes:

> The Class IP represents an IP address. (The "IP" file is already inside the project main folder);

> The classes mut be created following the UML structure in this pdf, you can add more classes and methods to the UML only if really needed (the teacher said we eventually would need to create more, so feel free to improve the UML as you keep going, but keep it as much simple as possible);

> The teacher gave us this: ip.Block block = new block(); but I have no idea what this is about (yet);

Rating criteria

Evaluation rubrics

ID	Description	Points
A	**Direct connection between two computers** The program allows you to connect computers directly and allows them to communicate with each other.	2
B	**Local Network** The program allows you to connect several computers to a switch (router) and allows them to communicate with each other (Function as a Switch).	2
C	**Two Networks Connected** The program allows you to connect two networks, each with its computers and switch, and allows them to communicate with each other.	2
D	**Multiple networks connected to a Router** The program allows you to connect two networks to a router, and allows them to communicate with each other.	2
E	**Two connected networks (efficient)** The program is configured in a similar topology to phase C but discards invalid requests before the TTL reaches 0.	2
F	**Knowledge Validation** The student knows how to explain the principles behind the code he has implemented. Javadocs and documentation.	until 10
G	**Unitary tests** The program has comprehensive unit tests of all relevant methods and scenarios.	until 2

Notice:

Rubrics are only evaluated upon presentation of the respective execution class with the main method.

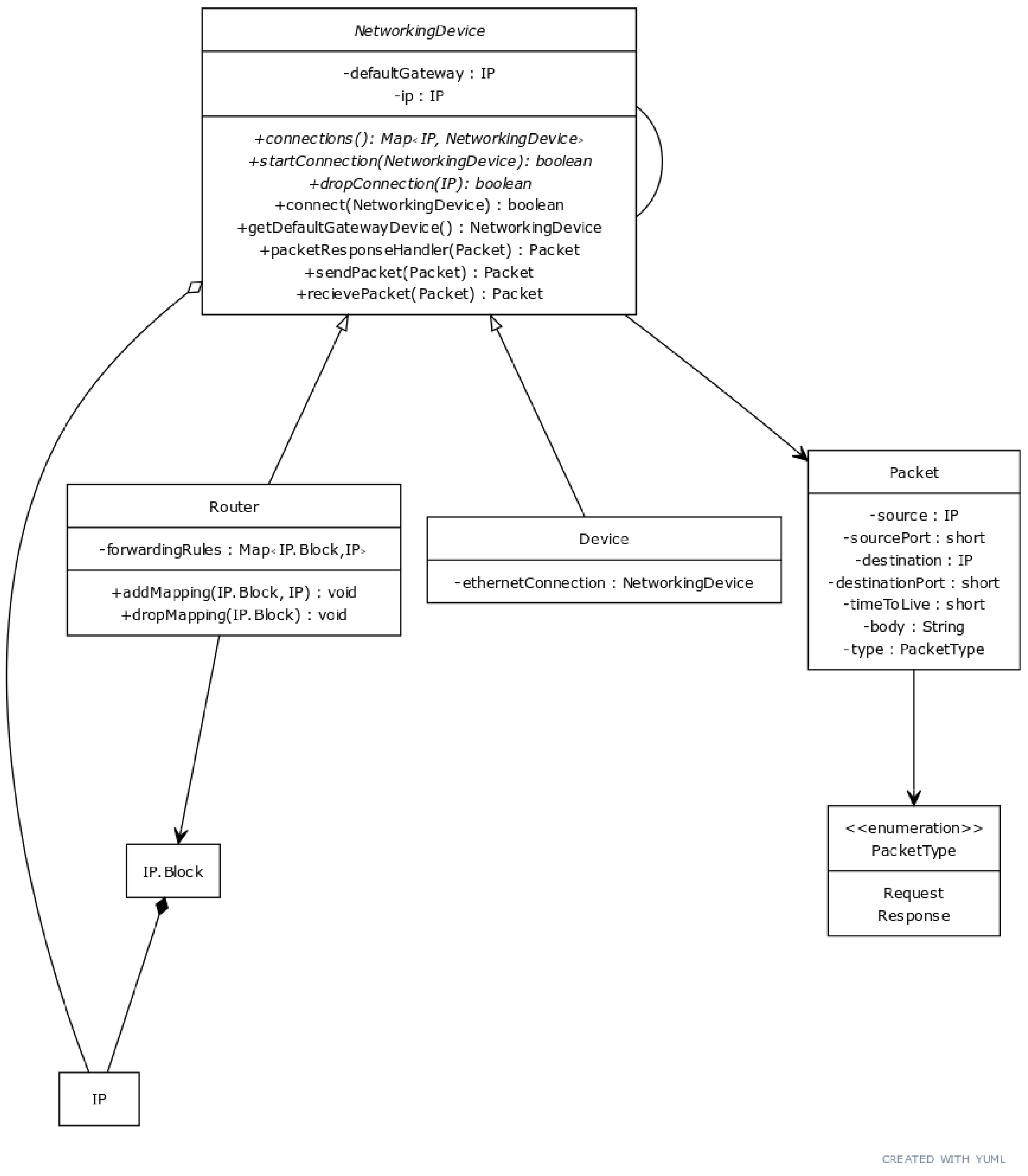
Sanctions

Each requirement has an associated score. The student scores 100% on a requirement when they implement and create the respective tests for a requirement and meet all style rules. If any good practice is not followed, a penalty occurs, according to the following table:

Problem	**Sanction**
The student does not comply with the conventions of Java names or use non-descriptive variable/method/class names.	Until10%
The student makes an incorrect use of whitespaces.	Until 5%
IMPORTANT The student does not document the code with JavaDocs or the quality of the JavaDocs/Comments is insufficient.	Until 20%
The student makes an incorrect use or inconsistent indentation. Note: the string used for indentation is chosen by the student, but must be consistent throughout the project.	Until 5%
The student makes other mistakes that are considered bad practices.	Until 10%
The student delivers code with performance issues (high complexity or excessive memory usage).	Until 25%
The student delivers code that only partially satisfies the requirement/use case or causes errors in certain use cases.	Until 100%

Implementation

UML Suggestion (more classes or methods can be added if needed)


NetworkingDevice
-defaultGateway : IP
-ip : IP
+connections(): Map<IP, NetworkingDevice>
+startConnection(NetworkingDevice): boolean
+dropConnection(IP): boolean
+connect(NetworkingDevice) : boolean
+getDefaultGatewayDevice() : NetworkingDevice
+packetResponseHandler(Packet) : Packet
+sendPacket(Packet) : Packet
+recievePacket(Packet) : Packet
Router
-forwardingRules : Map<IP.Block,IP>
+addMapping(IP.Block, IP) : void
+dropMapping(IP.Block) : void
Device
-ethernetConnection : NetworkingDevice
Packet
-source : IP
-sourcePort : short
-destination : IP
-destinationPort : short
-timeToLive : short
-body : String
-type : PacketType
<<enumeration>>
PacketType
Request
Response
IP.Block
IP
CREATED WITH YUML

Job Specification

IP

An IP is a 32-bit positive integer, represented in a format of four 8-bit segments, separated by periods:

###.###.###.###

An IP can belong to a block.

Block of IPs (IP.Block)

A block of IPs starts at a given IP and ends at another IP. A block is usually represented using CIDR notation.

The notation consists of a number in IP format followed by a slash and an integer from 0 to 32:

###.###.###.###/##

The IP represents the IP where the block is inserted.

The number represents how many bits of the IP are common to all IPs in the block.

In the case of 32, all bits are common in the block, so it is a block of 1 IP. .

In the case of 31, all but the last one are common, so it's a block of 2 IPs.

In the case of 30, all but the last two are common, so it's a block of 4 IPs.

…
In the case of 0, no number is common, being the block of all existing IPs.

Check CIDR calculator.

The most common CIDR blocks are /24, /16 and /8.

The block /24 is used in our home networks, and in the case of 192.168.0.0/24 it represents all IPs from 192.168.0.0 to 192.168.0.255.

The block /16 is used in larger LANs. 192.168.0.0/16 represents all IPs from 192.168.0.0 to 192,168,255,255.

The block /18 is used on the largest local area networks. 10.0.0.0/8 represents all IPs from 10.0.0.0 to 10,255,255,255.

Code for IPs and IP Blocks is provided.

NetworkingDevice

A network device is a device that can be connected to other network devices and that can handle packet transmission. A network device is addressed through its IP.

A network device communicates directly with the connected devices. When the network device you want to communicate with is not directly connected, it passes the request to its *default gateway*, if it exists.

Packet

A packet is a message sent by one network device to another network device. The receiver is not necessarily the intended receiver. Some network devices will be able to forward the packet to the desired receiver.

For performance reasons, the forwarding stops after X attempts, initially 128.

A packet is either a request or a response. To avoid an infinite packet cycle, a network device should only respond to requests.

Device

A device is an end network device. That is, it can represent a computer, a mobile phone or a server. It can respond to requests with special logic, such as returning a web page.

Router

A router is a network device responsible for forwarding requests. A router is connected to several devices and tries to forward requests to the devices to which it is connected, but before passing the request to its default gateway, it accesses its list of mappings and translates the IP to which it wants to transmit the packet to another IP, if you find a block of IPs in which your destination IP is inserted.

For example, if it has the following mapping:

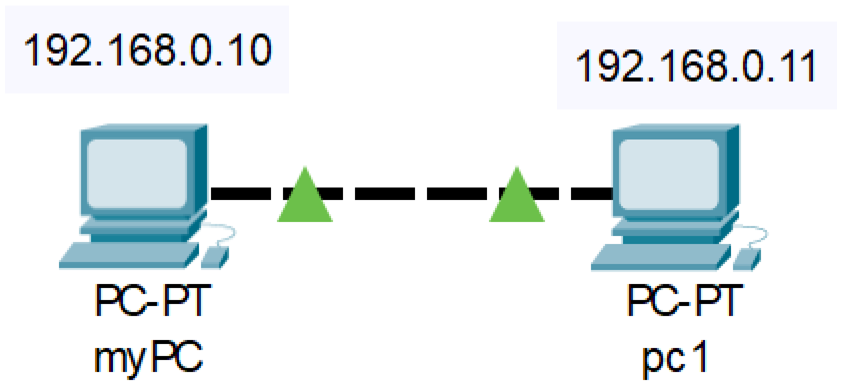
192.168.0.0/16 -> 80.124.15.3

And receive a packet destined for 192.168.0.33, it passes this packet to the network device connected with IP 80.124.15.3.

Implementation Stages

A - Direct connection between two computers

It is intended to make code to allow the following network:



In this aspect, it is necessary to implement the NetworkingDevice, Device and Packet classes.

NetworkingDevice

The NetworkingDevice uses the following logic to send a packet (sendPacket):

- Checks if the packet has a timeToLive (TTL) greater than 0;
- If the TTL is greater than 0, decrement it and continue;
- Attempts to deliver the packet to the recipient, traversing the set of its connections and invoking the recievePacket method on the destination connection, if it finds a connection whose IP matches the packet's destination;
- If it does not find a device with the intended IP, the handleDestinationNotFound method returns the evocation, which deals with the behavior when the destination is not found.

The NetworkingDevice uses the following logic to handle a package whose destination was not found (handleDestinationNotFound):

- If it has a default gateway, it returns the invocation of the method recievePacket on the device corresponding to the default gateway, otherwise it throws a RequestTimedOutException, made by us, or returns null, if you don't want/can implement exceptions.

The NetworkingDevice uses the following logic to receive a packet (recievePacket):

● If the IP destination is different from the IP itself, drop the packet, returning null or optionally throwing an exception.

● If the packet type is Request, returns the evocation resulto f the handlePacketResponse method.

● Otherwise, return the package itself.

A NetworkingDevice uses the following logic to respond to a packet (handlePacketResponse):

● Creates a new packet, whose source is the destination of the received packet (itself), whose destination is the source of the received packet, whose message is the message received with the following text before: `"I got your packet with the following contents: \n"` and whose type is Response.

One NetworkingDevice connects to another by invoking the startConnection method reciprocally, with the following code:

```
return this.startConnection(device) && device.startConnection(this);
```

Device

A Device uses the following logic to connect to a NetworkingDevice (startConnection):

- Save the NetworkingDevice in a field for later;
- Mark NetworkingDevice as your default gateway.

Device uses inverse logic to disconnect a connection.

Packet

A packet is represented as a String using the following code:

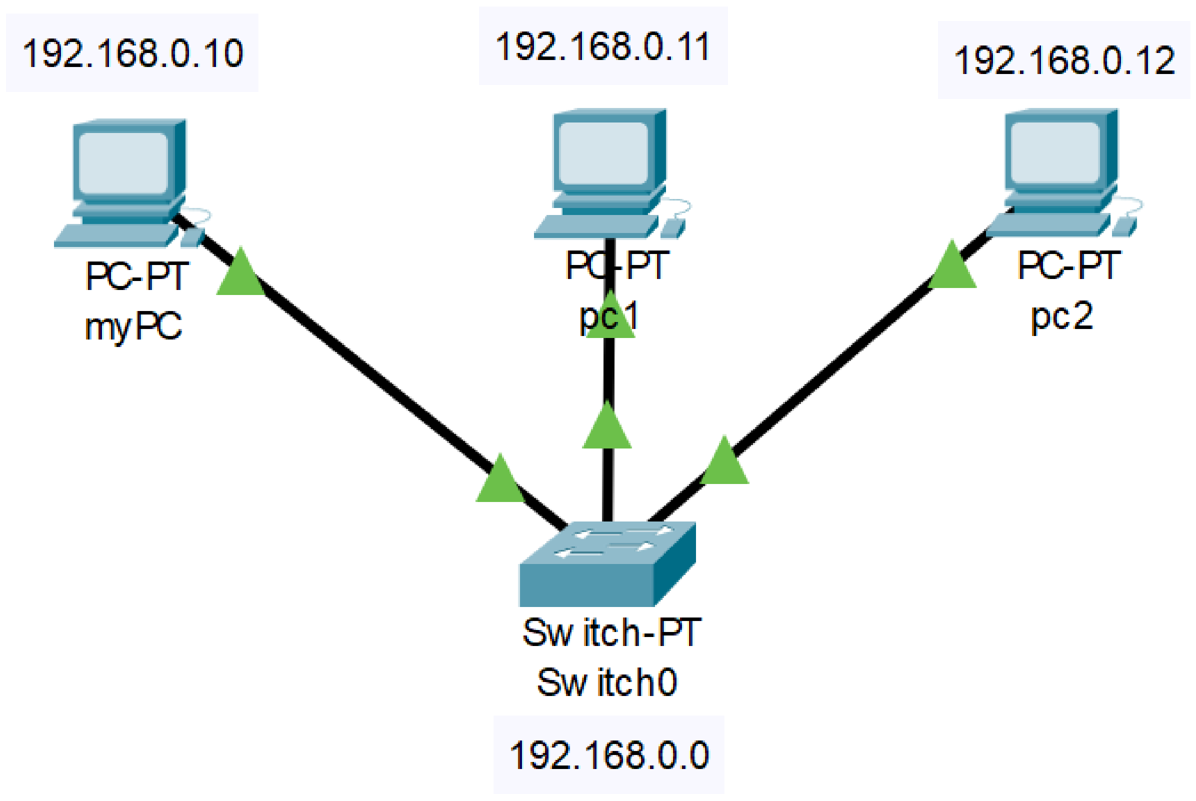
```
        "--------------------------------------------\n" +
        "From: " + source +
        "\nTo: " + destination +
        "\nTTL: " + timeToLive +
        "\nType: " + type.name() +
        "\nBody:\n" + body +
        "\n--------------------------------------------\n";
```

Functionality Test

Test if your network works by printing a packet sent to computer pc1. The message must go back and forth and must be changed accordingly.

B - Local Network

It is intended to make code to allow the following network:



In this aspect, it is necessary to implement the Router class. Our Router works as a Switch.

Router

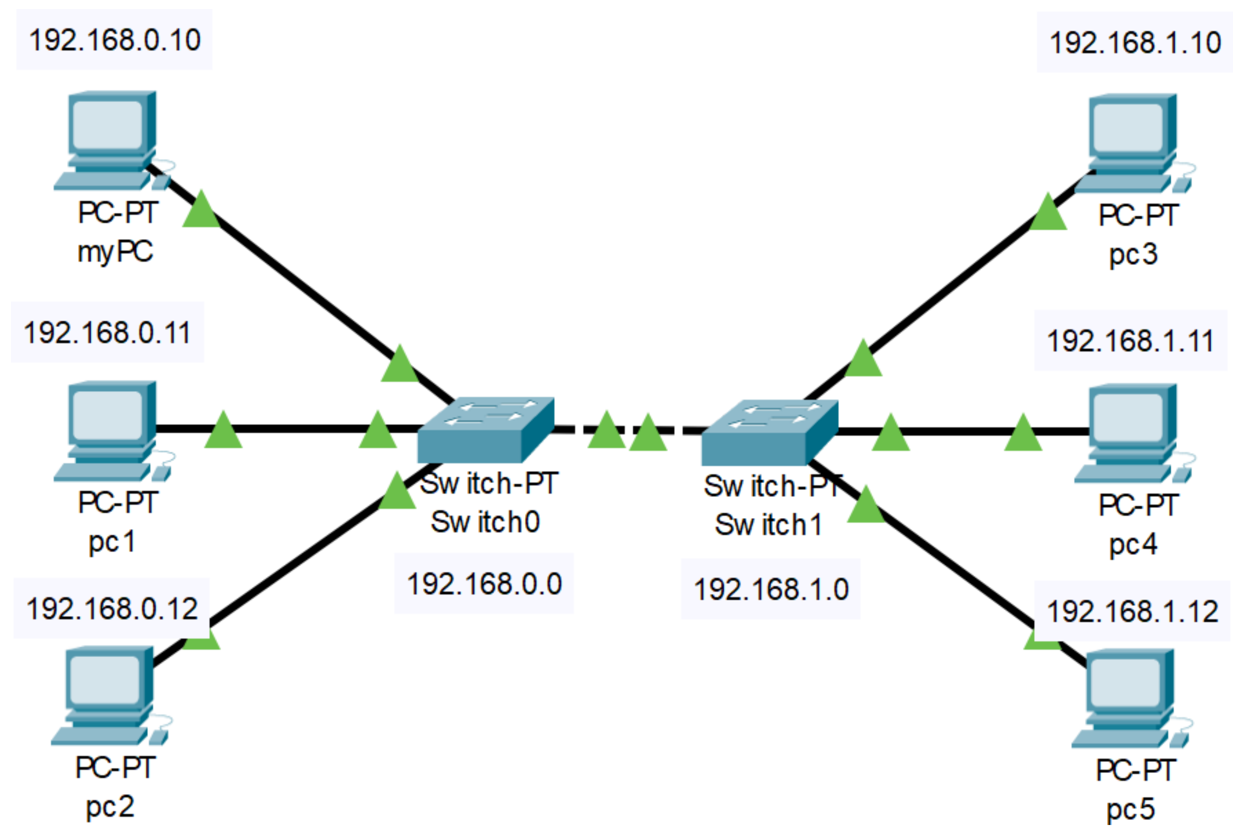
The Router works like a Device, but supports multiple connections. Must not have a *default gateway*. Functioning as Switch, it has no forwardingRules.

Functionality Test

Test if your network works by printing a packet sent to all computers and to the switch. The message must go back and forth and must be changed accordingly.

C - Two Networks Connected

It is intended to make code to allow the following network:



You must configure the *default gateways* of the switches as being one and the other, so that the computers can communicate with each other. The network must not fail if it receives a non-existent IP, and must therefore correctly implement the TTL (Time to Live) functionality.

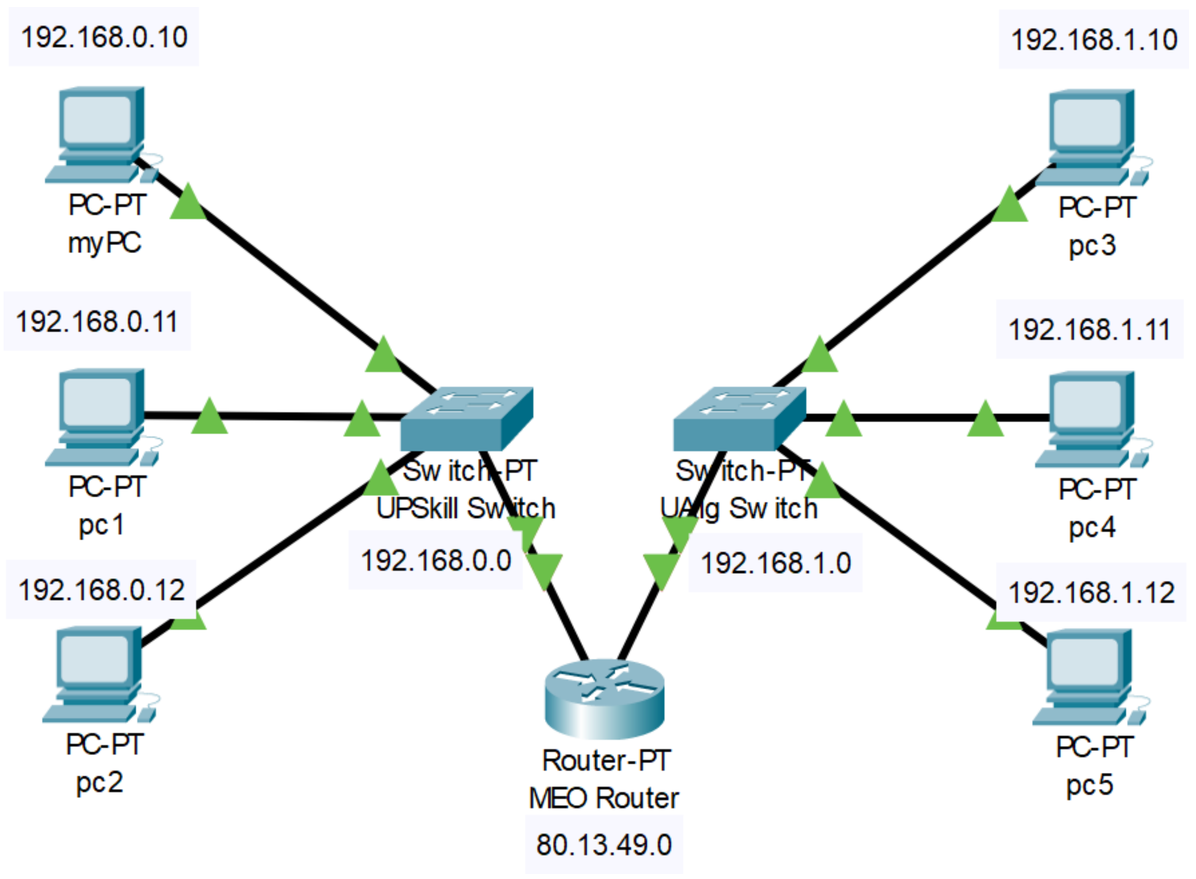
Functionality Test

Test if your network works by printing a packet sent to all computers and switches. The message must go back and forth and must be changed accordingly.

Also test if a packet sent to a non-existing IP is handled correctly.

D - Multiple networks connected to a Router

It is intended to make code to allow the following network:



You must configure the *default gateways* of the switches as the router. You can choose to keep the Switch in its class, having a Router class that extends its functionality, or choose to put the switches as Routers.

The Router has no default gateway.

The network must not fail if it receives a non-existent IP, and must therefore correctly implement the TTL (Time to Live) functionality.

Router

The router must implement IP mapping. If finds an IP that doesn't know, before passing the request to his default gateway, it must check the existing mappings and send the request to a given network device in case if it has a valid mapping.

The router must map all requests for IPs 192.168.0.# to the UPSkill Switch and all requests for IPs 192.168.1.# to the UAlg Switch.

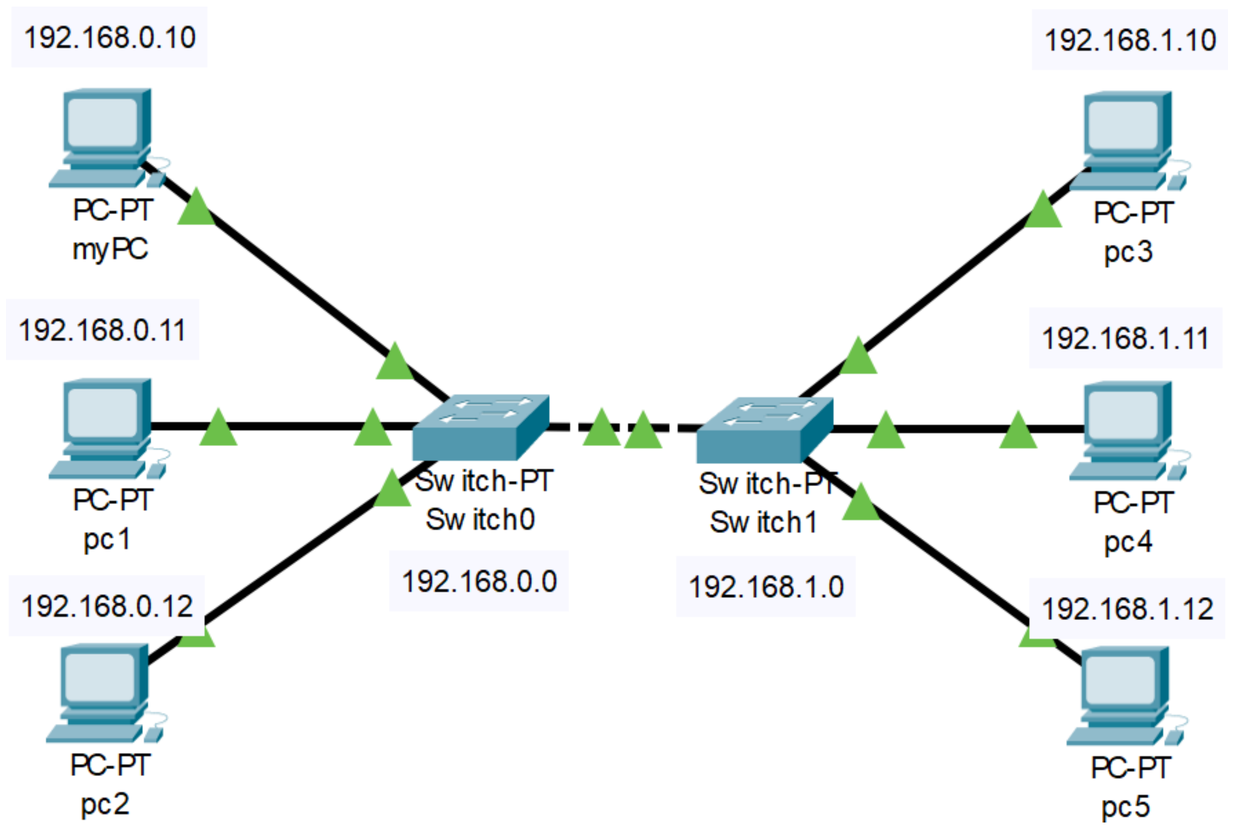
Functionality Test

Test if your network works by printing a packet sent to all computers and switches. The message must go back and forth and must be changed accordingly.

Also test if a packet sent to a non-existing IP is handled correctly.

E - Two Connected networks (efficient)

It is intended to make code to allow the following network:



As in stage C (two networks connected), however, the Switches must implement IP mapping instead of being reciprocal default gateways.

Functionality Test

Test if your network works by printing a packet sent to all computers and switches. The message must go back and forth and must be changed accordingly.

Also test if a packet sent to a non-existing IP is handled correctly.