MKC-PKS: Počítačové cvičení

Návrh sítě

Filip Paul

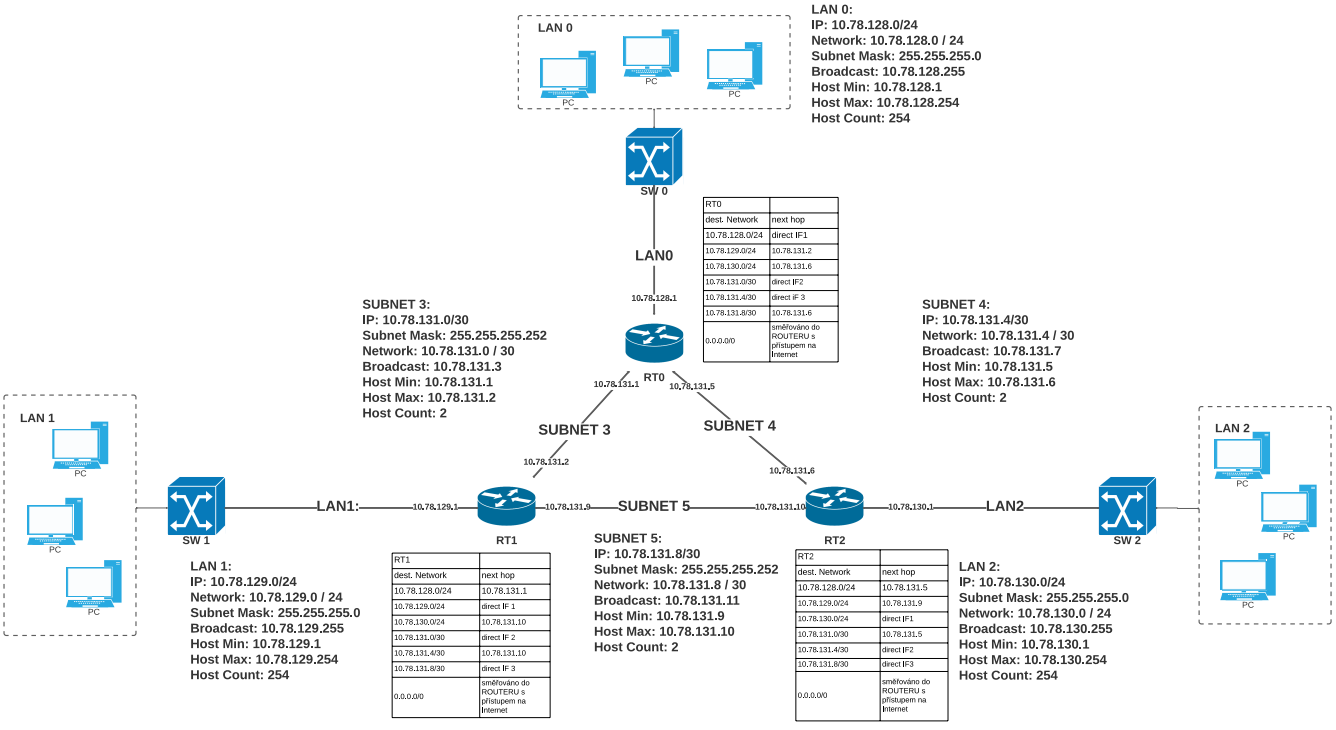
08.04.2022

Návrh sítě:

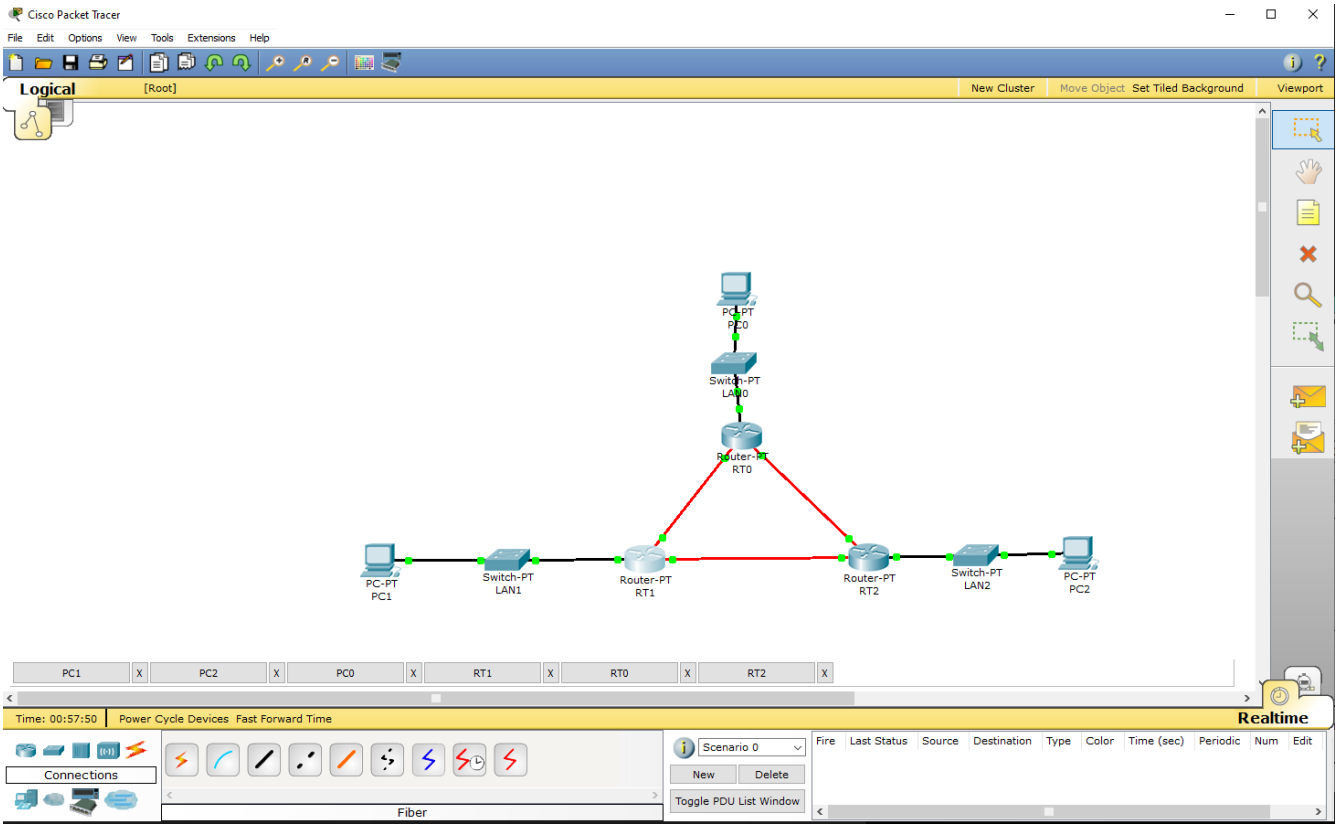
Vygenerovaná síť má následující adresní prostor:

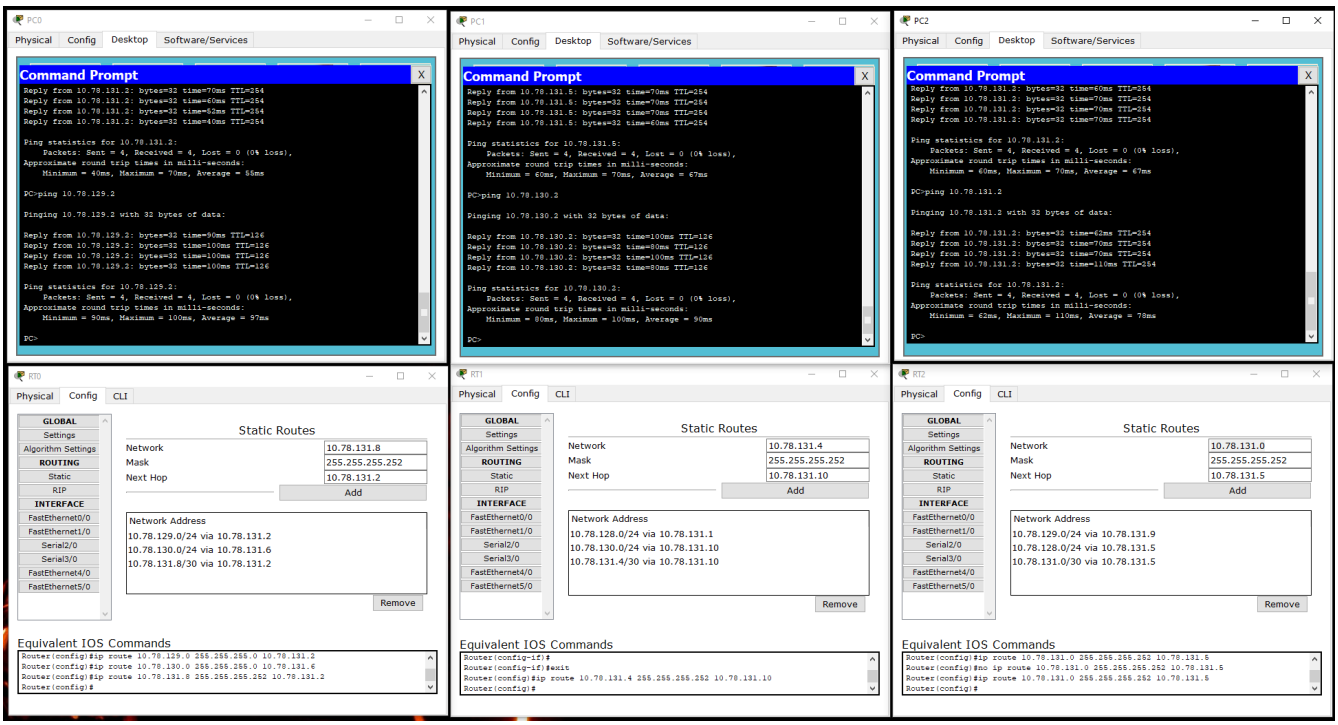
ID:	211538	Name:	Filip Paul
Address space:			
10.78.128.0/22			
Divide your IP address space into subnetworks as follows: a) make the address space in LAN0-LAN2 subnets as large as possible, b) make the address space in subnets connecting the routers as small as possible. Follow the convention that the gateway of the local networks (LAN0-LAN2) has the lowest possible IP address of the range. (Note: After this division thete will reserve for another subnets which could connect only two devices)			

Bohužel jsem se podíval na obrázek a rovnou jsem začal vytvářet síť podle postupu, který jsme dělali v prezenčních cvikách, což tak nějak neodpovídá zadání. Především tomu, že každá podsíť může využít až 31 IP adres. V mojem řešení je pro routery propojené napřímo dedikovaná menší síť. Celkově vznikne 6 podsítí. Pro výpočty parametrů sítí jsem využil vlastní python script, který si můžete zobrazit buď v mém GITHUB repozitáři zde nebo na konci dokumentu. V github repozitáři taky naleznete cisco_project.pkt soubor z PC simulace.



PC simulace:


Cisco Packet Tracer
File Edit Options View Tools Extensions Help
Logical
[Root]
New Cluster
Move Object
Set Tiled Background
Viewport
PC-PT PC0
Switch-PT LAN0
Router-PT RT0
PC-PT PC1
Switch-PT LAN1
Router-PT RT1
Router-PT RT2
Switch-PT LAN2
PC-PT PC2
PC1
PC2
PC0
RT1
RT0
RT2
Time: 00:57:50
Power Cycle Devices
Fast Forward Time
Realtime
Connections
Fiber
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time (sec)
Periodic
Num
Edit


Physical
Config
Desktop
Software/Services
Command Prompt
PC>ping 10.78.129.2
PC>ping 10.78.130.2
PC>ping 10.78.131.2
Physical
Config
CLI
Static Routes
Network
Mask
Next Hop
Add
Remove
Network Address
10.78.129.0/24 via 10.78.131.2
10.78.130.0/24 via 10.78.131.6
10.78.131.8/30 via 10.78.131.2
10.78.128.0/24 via 10.78.131.1
10.78.130.0/24 via 10.78.131.10
10.78.131.4/30 via 10.78.131.10
10.78.129.0/24 via 10.78.131.9
10.78.128.0/24 via 10.78.131.5
10.78.131.0/30 via 10.78.131.5
Equivalent IOS Commands

Přílohy:

calc.py

```
from myIPAdress import myIPAdress #import my custom class

print("\nK dispozici:")
myIPAdress("10.78.128.0/22").printInfo()

print("\nLAN 0")
myIPAdress("10.78.128.0/24").printInfo()

print("\nLAN 1")
myIPAdress("10.78.129.0/24").printInfo()

print("\nLAN 2")
myIPAdress("10.78.130.0/24").printInfo()

print("\nSUBNET 3")
myIPAdress("10.78.131.0/30").printInfo()

print("\nSUBNET 4")
myIPAdress("10.78.131.4/30").printInfo()

print("\nSUBNET 5")
myIPAdress("10.78.131.8/30").printInfo()
```

OUTPUT: K dispozici:
IP Address full format: 10.78.128.0/22
IP: 10.78.128.0
IP Bits: 00001010.01001110.10000000.00000000
Subnet Mask: 255.255.252.0 suffix: 22
Subnet Mask Bits: 11111111.11111111.11111100.00000000

Network: 10.78.128.0 / 22
Broadcast: 10.78.131.255
Host Min: 10.78.128.1
Host Max: 10.78.131.254
Host Count: 1022

LAN 0
IP Address full format: 10.78.128.0/24
IP: 10.78.128.0
IP Bits: 00001010.01001110.10000000.00000000
Subnet Mask: 255.255.255.0 suffix: 24
Subnet Mask Bits: 11111111.11111111.11111111.00000000

Network: 10.78.128.0 / 24
Broadcast: 10.78.128.255
Host Min: 10.78.128.1
Host Max: 10.78.128.254
Host Count: 254

LAN 1
IP Address full format: 10.78.129.0/24
IP: 10.78.129.0
IP Bits: 00001010.01001110.10000001.00000000
Subnet Mask: 255.255.255.0 suffix: 24
Subnet Mask Bits: 11111111.11111111.11111111.00000000

Network: 10.78.129.0 / 24
Broadcast: 10.78.129.255
Host Min: 10.78.129.1
Host Max: 10.78.129.254
Host Count: 254

LAN 2

IP Address full format: 10.78.130.0/24
IP: 10.78.130.0
IP Bits: 00001010.01001110.10000010.00000000
Subnet Mask: 255.255.255.0 suffix: 24
Subnet Mask Bits: 11111111.11111111.11111111.00000000

Network: 10.78.130.0 / 24
Broadcast: 10.78.130.255
Host Min: 10.78.130.1
Host Max: 10.78.130.254
Host Count: 254

SUBNET 3
IP Address full format: 10.78.131.0/30
IP: 10.78.131.0
IP Bits: 00001010.01001110.10000011.00000000
Subnet Mask: 255.255.255.252 suffix: 30
Subnet Mask Bits: 11111111.11111111.11111111.11111100

Network: 10.78.131.0 / 30
Broadcast: 10.78.131.3
Host Min: 10.78.131.1
Host Max: 10.78.131.2
Host Count: 2

SUBNET 4
IP Address full format: 10.78.131.4/30
IP: 10.78.131.4
IP Bits: 00001010.01001110.10000011.00000100
Subnet Mask: 255.255.255.252 suffix: 30
Subnet Mask Bits: 11111111.11111111.11111111.11111100

Network: 10.78.131.4 / 30
Broadcast: 10.78.131.7
Host Min: 10.78.131.5
Host Max: 10.78.131.6
Host Count: 2

SUBNET 5
IP Address full format: 10.78.131.8/30
IP: 10.78.131.8
IP Bits: 00001010.01001110.10000011.00001000
Subnet Mask: 255.255.255.252 suffix: 30
Subnet Mask Bits: 11111111.11111111.11111111.11111100

Network: 10.78.131.8 / 30
Broadcast: 10.78.131.11
Host Min: 10.78.131.9
Host Max: 10.78.131.10
Host Count: 2

myIPAdress.py

```
class myIPAdress():
    def __init__(self,ipaddr:str):
        self.ipaddr = ipaddr
        self.IP = ipaddr[:ipaddr.find("/")]
        self.IP_bits = self.get_IP_bits(ipaddr)
        self.subnet_mask, self.subnet_mask_bits, self.suffix = self.get_subnet_mask(ipaddr)
        self.broadcast = ""
        self.network = ""
        self.host_min = ""
        self.host_max = ""
        self.host_count = ""
```

```
        self.get_IP_range()

    def get_subnet_mask(self,ipaddr:str):
        pos = ipaddr.find("/")
        subnet = "1"* int(ipaddr[pos+1:]) + "0"*(32-int(ipaddr[pos+1:]))
        mask = ""
        mask_bits = ""
        for Bytes in range(0,31,8):
            mask +=   f"{int(subnet[ Bytes: Bytes+8], 2)}."
            mask_bits += f"{subnet[ Bytes: Bytes+8]}."

        return mask[:-1],mask_bits[:-1], int(ipaddr[pos+1:]) #remove last dot

    def get_IP_bits(self,ipaddr:str):
        ipaddr = ipaddr[:ipaddr.find("/")]

        IP_bits = ""
        for Bytes in ipaddr.split("."):
            IP_bits += f"{bin(int(Bytes))[2:].zfill(8)}."
        return IP_bits[:-1] #remove last dot

    def get_IP_range(self):
        current_IP_bytes = self.IP_bits.split(".")
        current_SUBNET_bytes = self.subnet_mask_bits.split(".")

        for IP_Byte,MASK_Byte in zip(current_IP_bytes,current_SUBNET_bytes):
            i = 0
            network_substract = 0
            broadcast_to_add = 0
            for IP_BIT,MASK_BIT in zip(IP_Byte,MASK_Byte):
                if MASK_BIT == "0":
                    broadcast_to_add += 2**(7-i)
                if MASK_BIT == "0" and IP_BIT == "1":
                    network_substract += 2**(7-i)
                i+=1

            self.broadcast += f"{int(IP_Byte,2)+broadcast_to_add-network_substract}."
            self.network += f"{int(IP_Byte,2) - network_substract}."



        self.broadcast = self.broadcast[:-1]
        self.network = self.network[:-1]
        last_byte_pos= self.broadcast.rfind(".")
        self.host_max = self.broadcast[:last_byte_pos+1] + f"{int(self.broadcast.split('.')[-1])
    -1}"
        last_byte_pos= self.IP.rfind(".")
        self.host_min = self.network[:last_byte_pos+1] + f"{int(self.network.split('.')[-1])+1}"
        self.host_count = 2**(32-self.suffix) - 2

    def printInfo(self):
        print("IP Address full format:", self.ipaddr)
        print("IP:", self.IP)
        print("IP Bits:", self.IP_bits)
        print("Subnet Mask:", self.subnet_mask, "suffix:", self.suffix)
        print("Subnet Mask Bits:", self.subnet_mask_bits)
        print("\nNetwork:", self.network, "/", self.suffix)
        print("Broadcast:", self.broadcast)
        print("Host Min:", self.host_min)
        print("Host Max:", self.host_max)
        print("Host Count:", self.host_count)
```