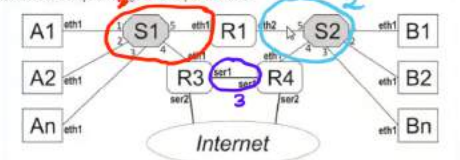
28/05

Tenha em consideração a figura 1 que ilustra o equipamento duma instituição Y que é necessário interligar através de IPv4 à Internet. A instituição possui dos departamentos diferentes, A e B. Os equipamentos referidos como **An** são *hosts* do departamento A e os equipamentos referidos como **Bn** são *hosts* do departamento B. Os equipamentos referidos como **S1** e **S2** são comutadores (*switches ethernet*) e os referidos por **R1**, **R3** e **R4** são encaminhadores (*routers*) IPv4. O *router* **R1** serve para interligar as redes dos dois departamentos e os routers **R3** e **R4** servem para interligar os departamentos através duma linha dedicada e também para interligar a instituição Y à Internet.



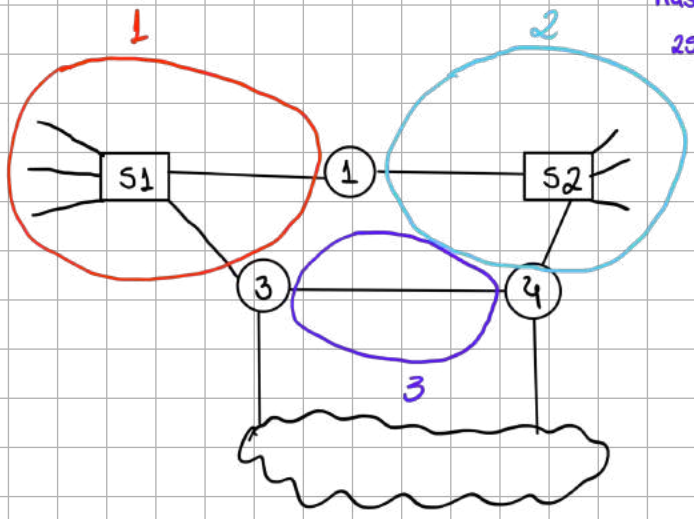
1. Tendo em consideração que a instituição Y tem apenas disponível uma rede classe C para o endereçamento de todos os equipamentos, defina um esquema de endereçamento que maximize o valor de **n**, i.e., que permita o maior número possível de *hosts* em cada sub-rede departamental (escolha um endereço IPv4 classe C a seu gosto diferente de 192.168.*.0)

m → nº de hosts

Classe C : 192.0.0.0 a 223.255.255.255

Para simplificar, escolhemos 200.0.0.0/24

↳ Máscara classe C
255.255.255.0



numa sub-rede:
o mais comum é atribuir o 1º endereço disponível a um router e o último endereço disponível ao outro router.

⇒ switches são invisíveis à camada de rede

End. Rede: 200.0.0.0

3 subredes
mínimo de bits pl representar 3 subredes

$2^2 = 4 > 3$ portanto, 2 bits.

200.0.0.0/24 → +2 bits

Máscara Subnetting: 26 bits → 255.255.255.11000000
255.255.255.192
(128+64)

00 → subrede 1
01 → subrede 2
10 → subrede 3
11 → salvaguarda pl expansão no futuro

2 bits subrede | 6 bits pl hosts

Host / Router	Endereço sub-rede	Endereço Interface	Endereço	Endereço completo
R1	00	eth 1	000001	200.0.0.00000001 200.0.0.1/26
R1	01	eth 2	000001	200.0.0.01000001 200.0.0.65/26
R3	00	eth 1	111110	200.0.0.00111110 200.0.0.62/26
R3	10	ser 1	000001	200.0.0.10000001 200.0.0.129/26
R4	01	eth 1	111110	200.0.0.01111110 200.0.0.126/26
R4	10	ser 1	000010	200.0.0.10000010 200.0.0.130/26
A1 → 2º disponível	00	eth 1	000010	200.0.0.00000010 200.0.0.2/26
Am → penúltimo disponível	00	eth 1	111101	200.0.0.00111101 200.0.0.61/26
B1	01	eth 1	000010	200.0.0.01000010 200.0.0.66/26
Bm	01	eth 1	111101	200.0.0.01111101 200.0.0.125/26

na sub-rede 3:

Sendo R3 e R4 não há hosts disponíveis, portanto, podemos atribuir o 1º endereço disponível a um e o 2º endereço disponível ao outro.

1º host: 1º endereço disponível, pois o 1º foi atribuído a um router.

último host: penúltimo endereço disponível, pois o último foi atribuído a um router.

2. Sabendo que os dois departamentos têm que ter interligação entre si e à Internet, complete as tabelas de encaminhamento manual/estático IPv4 para **A1**, **R1** e **R4** (a ordem das entradas numa tabela é irrelevante; escreva os endereços no formato CIDR):

Tabela de encaminhamento de R4 R4

	Rede/Sub-rede Destino	Próximo *Hop*	Interface de saída
1º	0.0.0.0	128.20.0.6/30	ser2
2º	128.20.0.4/30	128.20.0.5/30	ser2
Subrede 1	200.0.0.0126	200.0.0.129	ser 1
Subrede 2	200.0.0.64126	200.0.0.126	eth 1
Subrede 3	200.0.0.128126	200.0.0.130	ser 1

4

1º quando não sabe pl onde encaminhar, default é o router do ISP.
2º comunicar ao router do ISP que fornece o serviço
↳ subrede R4 e router ISP, conectividade com exterior

R4 pl subrede 1: ou por R1 ou por R3
por R1 dá 1 salto a mais
melhor por R3 (+ rápido)
mas a nível de rede dava no mesmo

R4 pl subrede 2:
Próximo hop é ele mesmo, já está na rede que é destinado
Gateway: 0.0.0.0 pois não se usa gateway já que não encaminha entre redes

R4 pl subrede 3: mesma explicação que a anterior

Tabela de encaminhamento de A1

Rede/Sub-rede Destino	Próximo *Hop*	Interface de saída
200.0.0.0/26	200.0.0.2	eth 1
0.0.0.0/0	200.0.0.62	eth 1

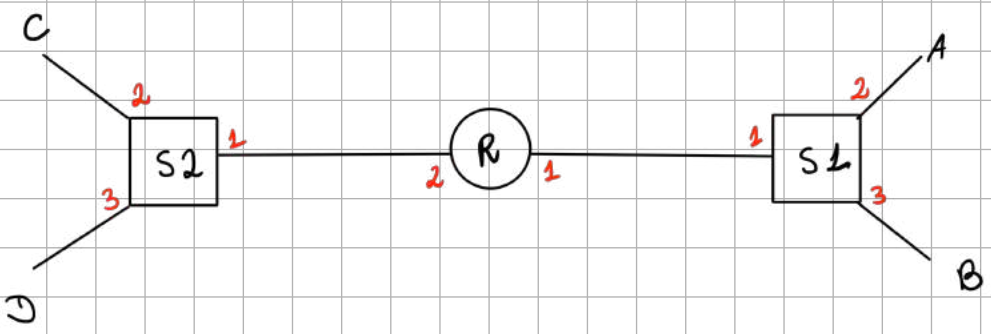
host tem sempre apenas duas entradas
↳ acesso local
↳ default

default:
pl R1 adiciona + um salto pl ir pl internet
portanto deve ser pl R3

Tabela de encaminhamento de R1

Rede/Sub-rede Destino	Próximo *Hop*	Interface de saída
0.0.0.010	200.0.0.126	eth 1
200. 0.0.0126	200. 0.0. 1	eth 1
200. 0.0. 64126	200. 0. 0. 65	eth 2
200. 0. 0. 128126	200. 0.0. 62	eth 1

default e sub-rede 3: indiferente escolher entre R3 e R4



IP
destino: IPC
origem: IP A
MAC
destino R1
origem A

A envia pacote para C

S1 guarda a partir da porta 2

	MAC	Interface	TTL
S1	MAC A	2	60
S2	MAC R2	1	60

tempo que guarda entrada, decrementa e se chegar a 0, apaga a entrada.
Se já existir a entrada, dá refresh e reinicia o TTL

S1 procura por entrada R1, não existe
flooding, R1 e B recebem
B descarta

S2 flooding
D descarta
C desencapsula, vê que é ICMP e envia resposta
S2 salva entrada MAC C, já tem entrada p/ R2, envia p/ R2 diretamente

S1 recebe de R1, salva entrada e envia p/ A pois já tinha sua entrada salva.

	MAC	Interface	TTL
S1	MAC A	2	60
S1	MAC R1	1	60
S2	MAC R2	1	60
S2	MAC C	2	60

Flooding

Flooding means that the switch sends the incoming frame to all occupied and active ports (except for the one from which it was received). In essence, flooding is when a switch pretends to be a hub. There are two basic reasons why a switch will flood a frame.

1. When the switch receives a broadcast, it has no choice but to continue the broadcast. Protocols like ARP and DHCP (among others) rely on these broadcasts for their basic function. The following diagram is an example of what an Ethernet frame header might look like as a broadcast.

[FF:FF:FF:FF:FF:FF]	[02:60:8c:12:34:56]	[0806]
Destination	Source	ARP

2. When the switch receives a frame dedicated for a particular destination but that destination does not have an entry in the MAC Address Table, the switch has no choice but to flood the frame. The goal of this flood is that the device using the MAC address in the destination of the frame will receive the flood and respond to the message. If that device responds, then the switch can learn their MAC address and map it to the port into which the message arrives. The following diagram is an example of what an Ethernet frame header might look like. Notice that the destination MAC address does not match the MAC Address Table above.

[00:02:67:80:5c:1a]	[02:60:8c:12:34:56]	[0800]
Destination	Source	IP

4. Sabendo que o MTU (*Maximum Transmission Unit*) da rede dedicada entre **R3** e **R4** é de 420 bytes, **R3** tem que fragmentar um pacote IPv4 que recebeu de **A1**, com um total de 900 bytes, por forma a enviar os fragmentos para **R4**. O pacote IPv4 original recebido de **A1** tem o seguinte cabeçalho (o símbolo "?" indica que o valor destes campos é irrelevante neste exercício):

Ver = 4	IHL = 5	Type of Service = ?	Total Length = 900	
Identification = 33333			Flags=000	Fragment Offset = 0
Time To Live = 5		Protocol = ?	Header Checksum = ?	
Source IP Address = ?				
Destination IP Address = ?				

5

Preencha os campos dos seguintes cabeçalhos dos pacotes IP resultantes do processo de fragmentação do pacote original e que serão enviados a **R4**:

Ver = 4	HL = [ ]	Type of Service = ?	Total Length = [ ]	
Identification = [ ]			Flags=[ ]	Fragment Offset = [ ]
Time To Live = [ ]		Protocol = ?	Header Checksum = ?	
Source IP Address = ?				
Destination IP Address = ?				

MTU : 420 bytes
cabeçalho IPv4 : 20 bytes
dados : 420 - 20 = 400 bytes

pacote total de 900 bytes inclui cabeçalho
880 a nível de dados

offset: destino reagrupa fragmentos pl saber pacote original, precisa saber a ordem pl reconstruir

Fragmento 1 880 - 400 = 480 bytes → não vai ser o último fragmento

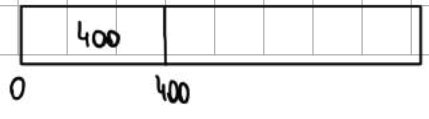
Ver = 4	HL = [5]	Type of Service = ?	Total Length = [420]	
Identification = [33333]			Flags=[001]	Fragment Offset = [0]
Time To Live = [4]		Protocol = ?	Header Checksum = ?	
Source IP Address = ?				
Destination IP Address = ?				

4 x 5 = 20 bytes do cabeçalho

id - mesmo que pacote original → identificar fragmentos do mesmo pacote

offset : onde ficam a partir do original

dtl original = 5
dtl de todos fragmentos = 5 - 1 = 4



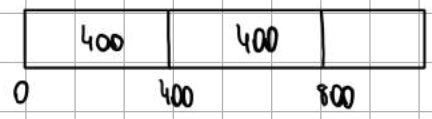
Fragmento 2 — 480 − 400 = 80 bytes → ainda há + fragmento

Ver = 4	HL = [5]	Type of Service = ?	Total Length = [420]
Identification = [33333]		Flags=[001]	Fragment Offset = [400]
Time To Live = [4]		Protocol = ?	Header Checksum = ?
Source IP Address = ?			
Destination IP Address = ?			

offset → palavras de 8 bytes

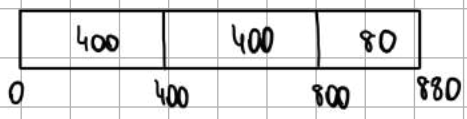
offset de 300 = 800 bytes

⇒ melhor meter em bytes e não dividir por 8 p/ não errar cálculos e complicar fragmentação em casos não divisíveis por 8



Fragmento 3 — 80 bytes < 400 → último fragmento

Ver = 4	HL = [5]	Type of Service = ?	Total Length = [100]
Identification = [33333]		Flags=[000]	Fragment Offset = [800]
Time To Live = [4]		Protocol = ?	Header Checksum = ?
Source IP Address = ?			
Destination IP Address = ?			

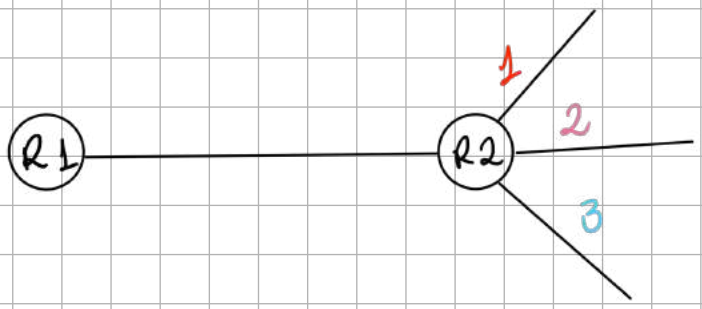


Inicial : 900 bytes

Fragmentos : 420 + 420 + 300 = 940 bytes

Supernetting

=> Arranjar maneira de ter menos entradas na tabela



1 : 200.0.0.16/28 — 16 : 00010000

2 : 200.0.0.32/28 — 32 : 00100000

3 : 200.0.0.48/28 — 48 : 00110000

iguais

=> primeiros 26 bits são iguais

Destination	Next hop	Interface
200.0.0.0/26	R2	eth 1

=> converter 3 entradas em apenas uma.