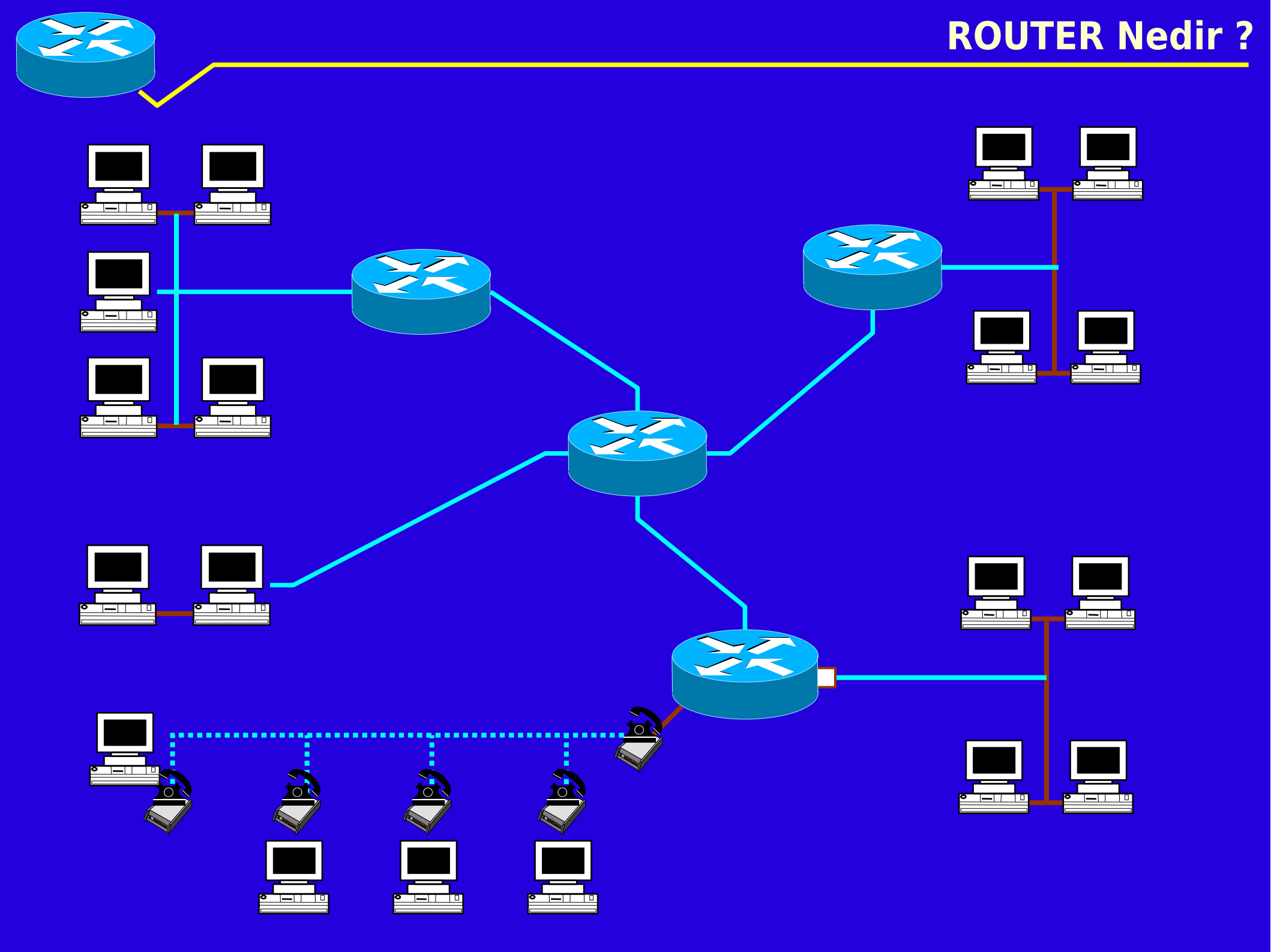
Linux Router Konsepti
Gelişmiş Yönlendirme Kabiliye

Serdar KÖYLÜ
Gelecek A.Ş

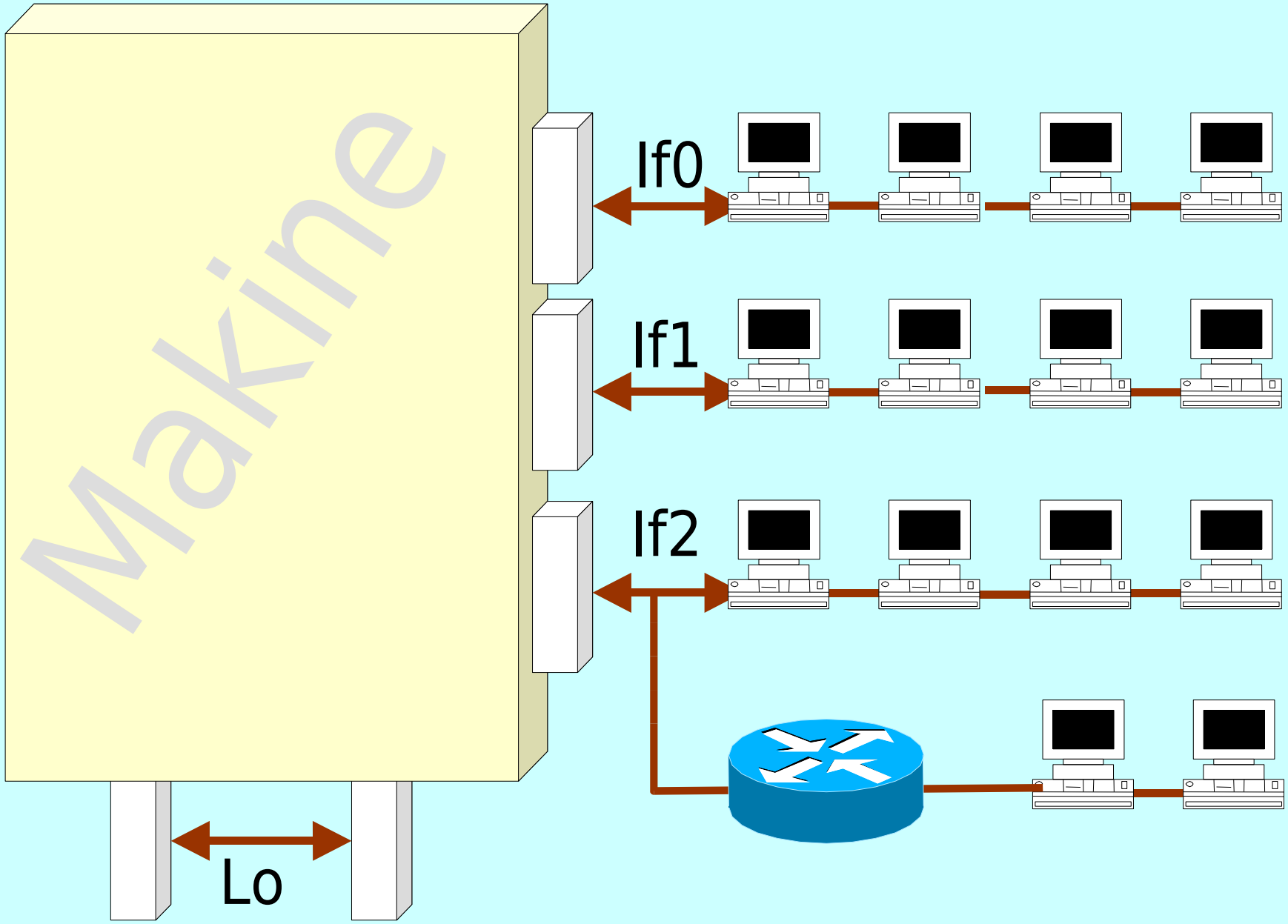
ROUTER Nedir ?

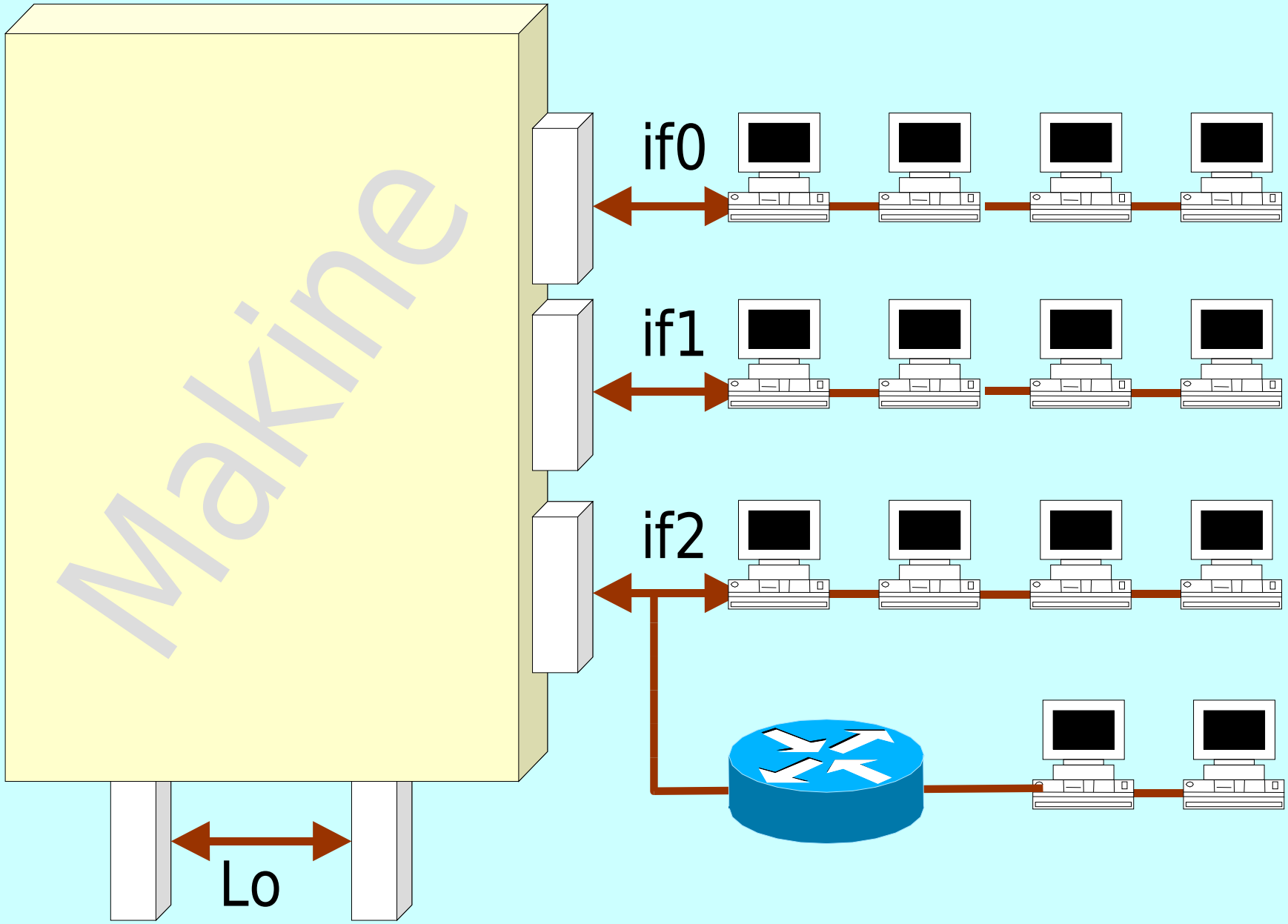


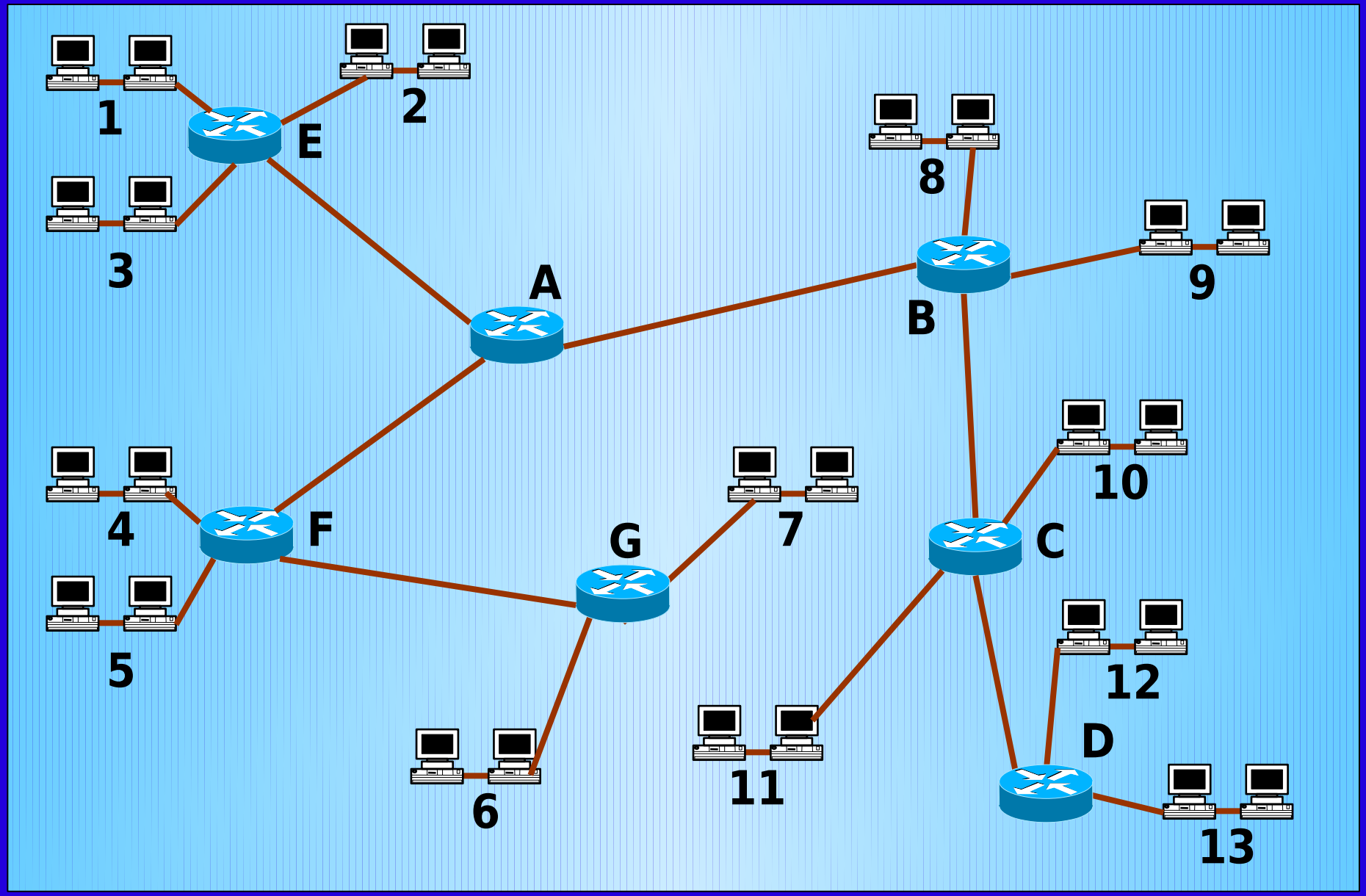
Ağ üzerindeki birimlerin birbirlerine veri göndermek için ihtiyaç duydukları fiziksel iletişim cihazları "Arabirim" "INTERFACE" olarak tanımlanır.

Ethernet Arabirimleri, PSTN, DSL Modemler,

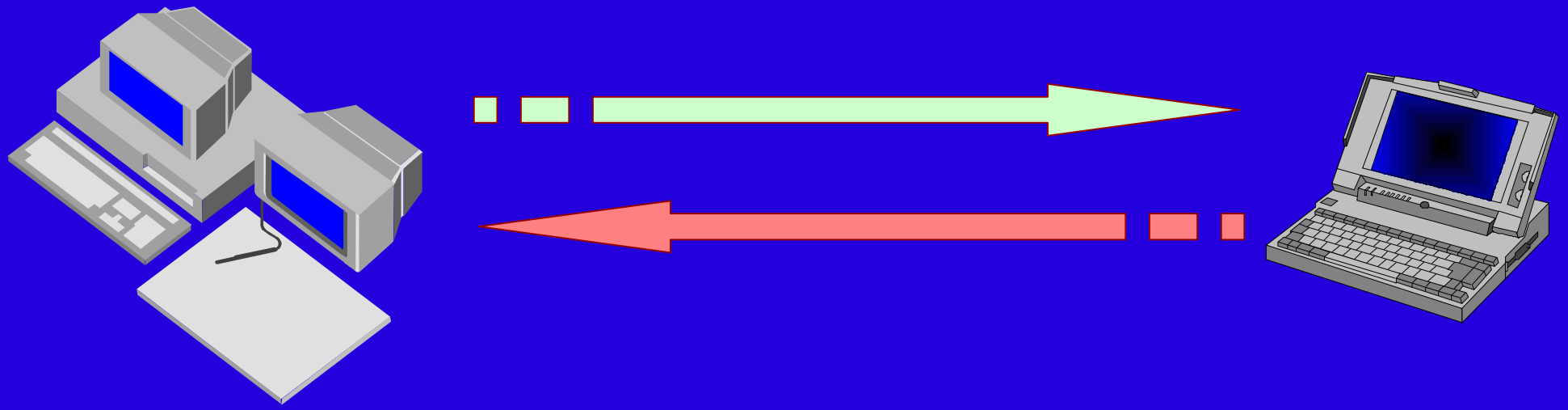
V35, T1/E1 Arabirimleri, Bluetooth gibi Telsiz Adapterler


If0
If1
If2
Lo
Makine

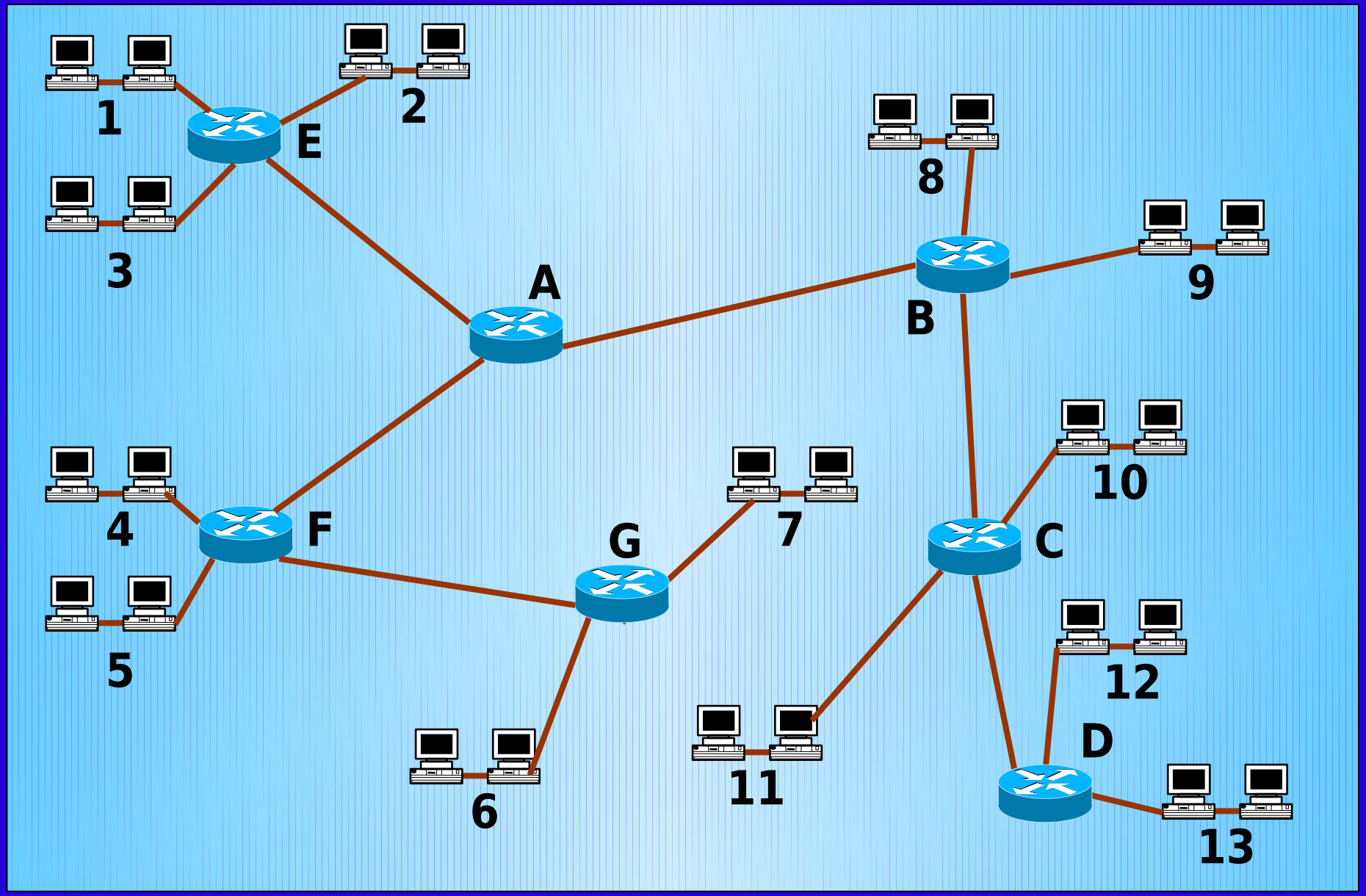

Makine
if0
if1
if2
Lo


1
2
E
3
A
8
B
9
4
F
G
7
10
C
5
12
D
6
11
13

Temel Kavramlar – Bantgeniş



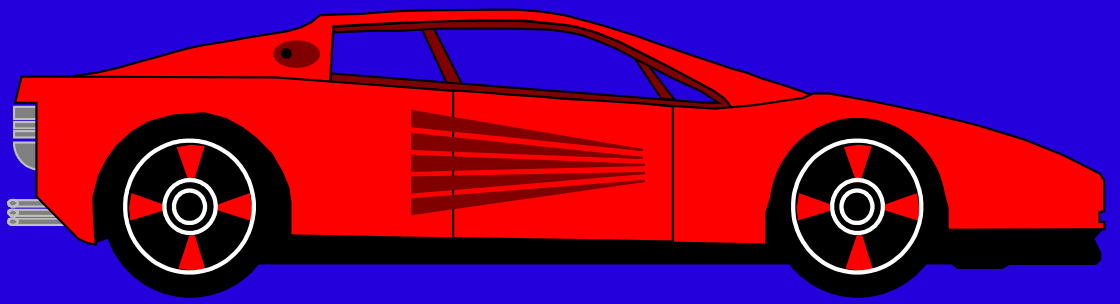
Temel Kavramlar – METR

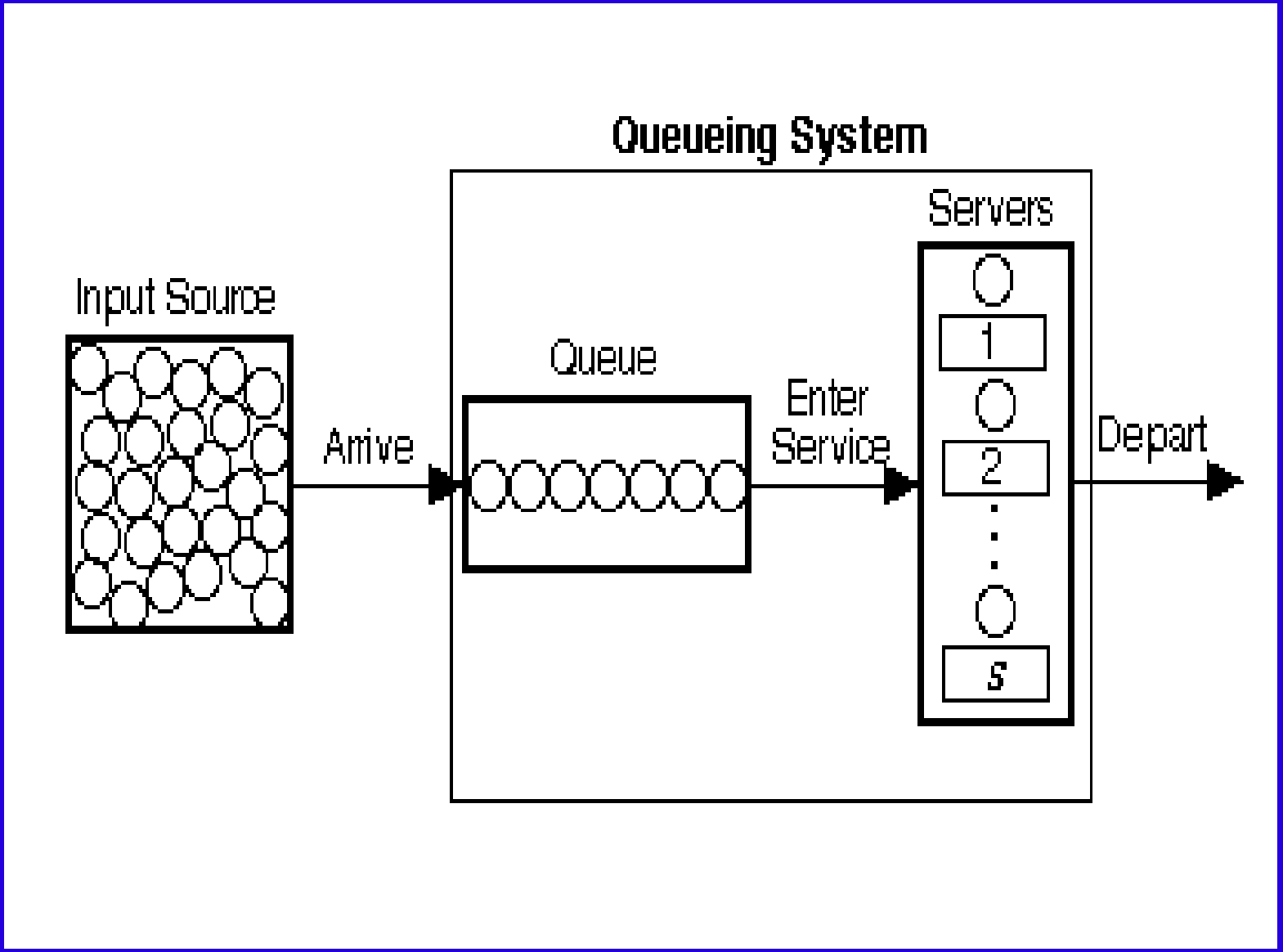


Bir defada iletilebilecek maksimum paket boyu

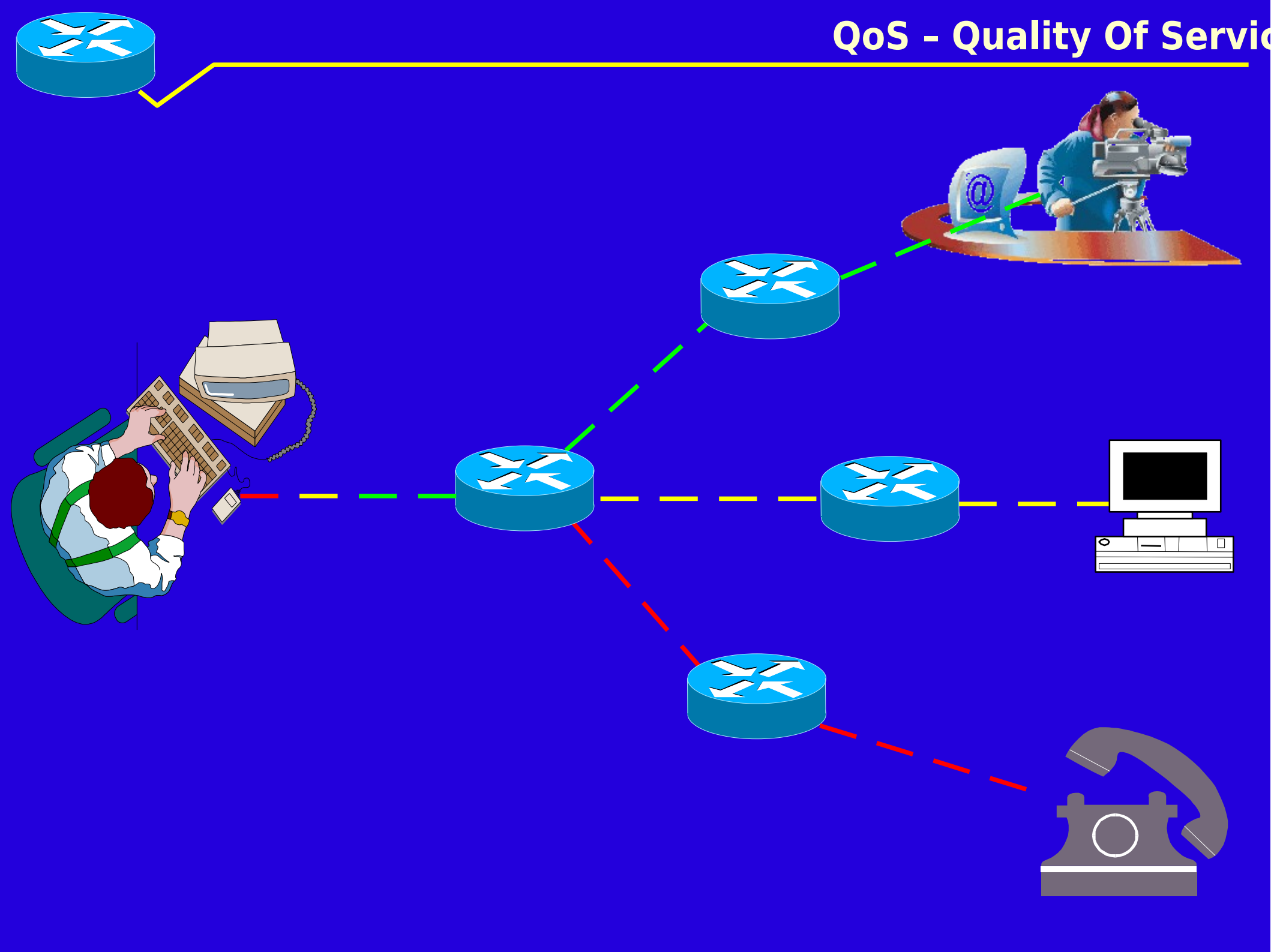
Layer 1 & 2- Ethernet, ppp vs.
Layer3 – IP paket boyu

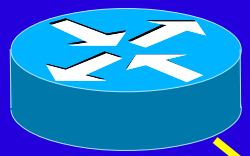
IP cihazları 576 octetlik paketleri garantilemelidirler.




Queueing System
Servers
Input Source
Queue
Arrive
Enter Service
Depart
1
2
s

QoS – Quality Of Servic



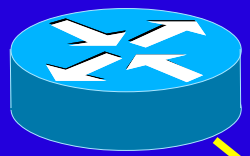


Bantgenişliği problemleri

- Ortak kullanılan hatlar.
- Kuyruk kavramı
- Latency ve Lag..
- Öncelik tanımlama.

Throughput sorunu

- İşlenebilecek veri miktarı.
- DoS durumu.
- Load balancing.



Uzak yerleşimler.

- İki farklı bölge arası trafik
  - Yerel ağları birleştirme
  - Güvenlik sorunları

Güvenlik sorunları.

- Erişim kısıtlama – Firewall konsepti
- Güvenli erişim
- DoS saldırıları.

Linux Firewall Konsepti

Netfilter;

Linux 2.4+ serisi kerneller için firewall altsistemi..

IPTables;

Bu altsistemi yönetmeyi sağlayan kullanıcı seviyesi program..

Netfilter'in kabiliyetleri

Packet filter;

IP paketlerinden istenilen kriterlere uygun olanlar durdurulabilir.

Full NAT;

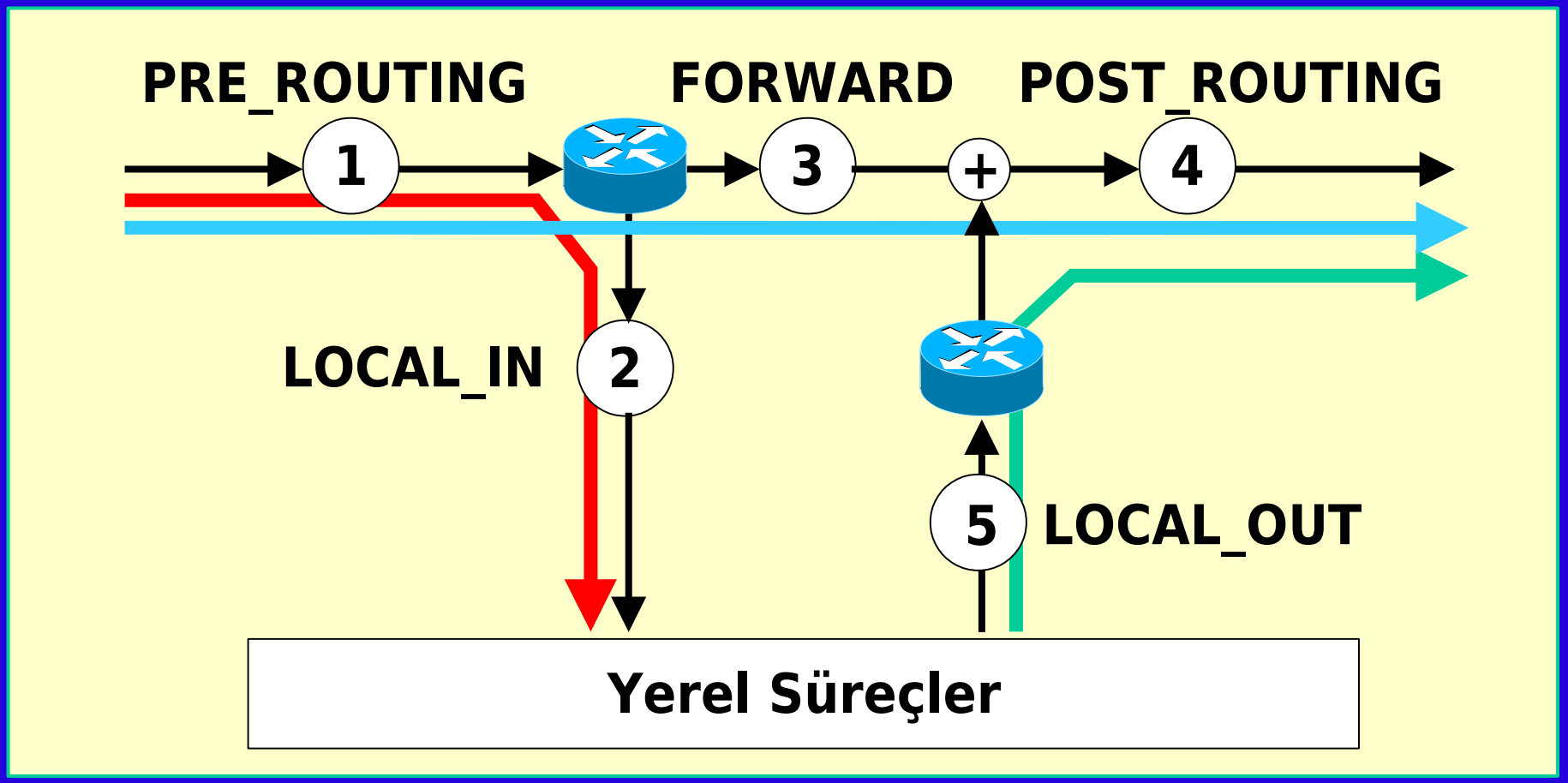
Gelen paketler, orijinal hedeflerinden farklı yerlere yönlendirilebilir, farklı adresten geliyormuş gibi gösterilebilir

Connection Tracking;

Statefull çalışabilme yetisi

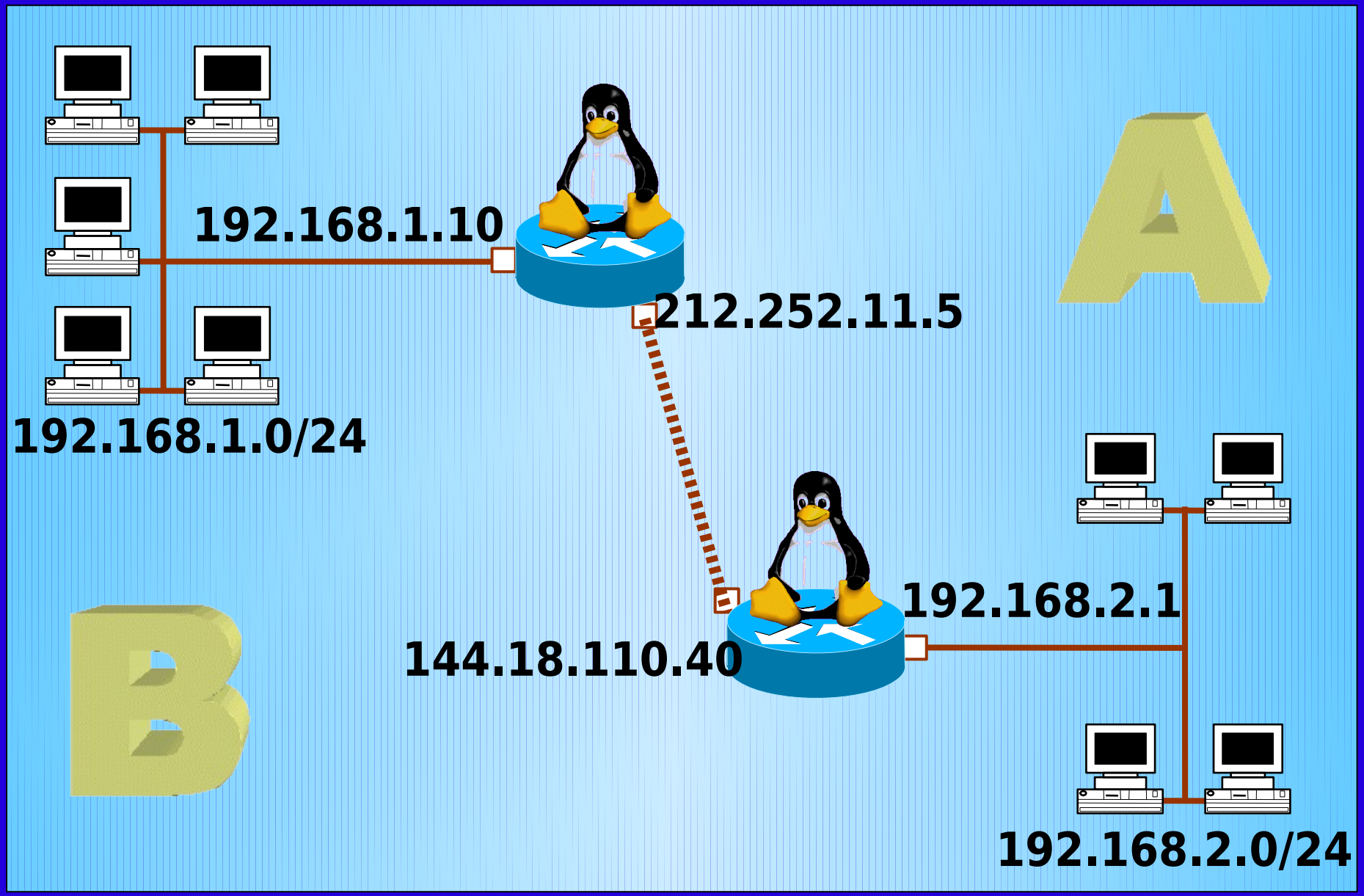
Packet Mangling

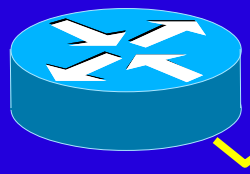
Paketleri işaretleyebilme ve özel alanlarını düzenleyebilme kabiliyeti.




```
# iptables -A PREROUTING -t mangle \
        -p tcp --dport 80 -j MARK --set-mark 1

# ip rule add fwmark 1 table webdata
# ip route add default via 195.255.51.3 dev ppp1 table webdata
```

Tunneling Fonksiyonla





Router A ve B

```
insmod ipip
insmod new_tunnel
```

Router A

```
ifconfig tunl0 192.168.1.10 pointopoint
    144.18.110.40
route add -net 192.168.2.0 netmask 255.255.255.0
    dev tunl0
```

Router B

```
ifconfig tunl0 192.168.2.1 pointopoint 212.252.11.5
route add -net 192.168.1.0 netmask 255.255.255.0
    dev tunl0
```

Tüneli kapatma

```
ifconfig tunl0 down
```

IP-on-IP Avantajları

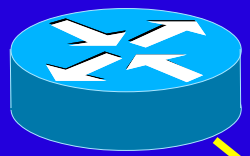
- Basittir.
- Dial-Up kullanıma elverişlidir.
- Az sistem kaynağı gerektirir.

IP-on-IP Dezavantajları

- Sadece Linux ile kullanılabilir.
- Bazı routerlar ile uyumsuz olabildiği rapor edilmiştir. Sebep, IPv6 ve Multicast desteklenmez.

Çözüm

GRE Tunneling protocol



Router A

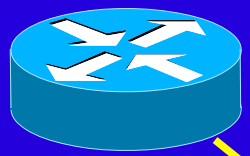
```
ip tunnel add istanbul mode gre remote 212.252.11.5
     \
     local 144.18.110.40 ttl 255
ip addr add 192.168.10.1 dev istanbul
ip route add 192.168.1.0/24 dev istanbul
```

Router B

```
ip tunnel add ankara mode gre remote 144.18.110.40
     \
     local 212.252.11.5 ttl 255
ip addr add 192.168.11.1 dev ankara
ip route add 192.168.2.0/24 dev ankara
```

Modülü ise...

```
modprobe ip_gre
```



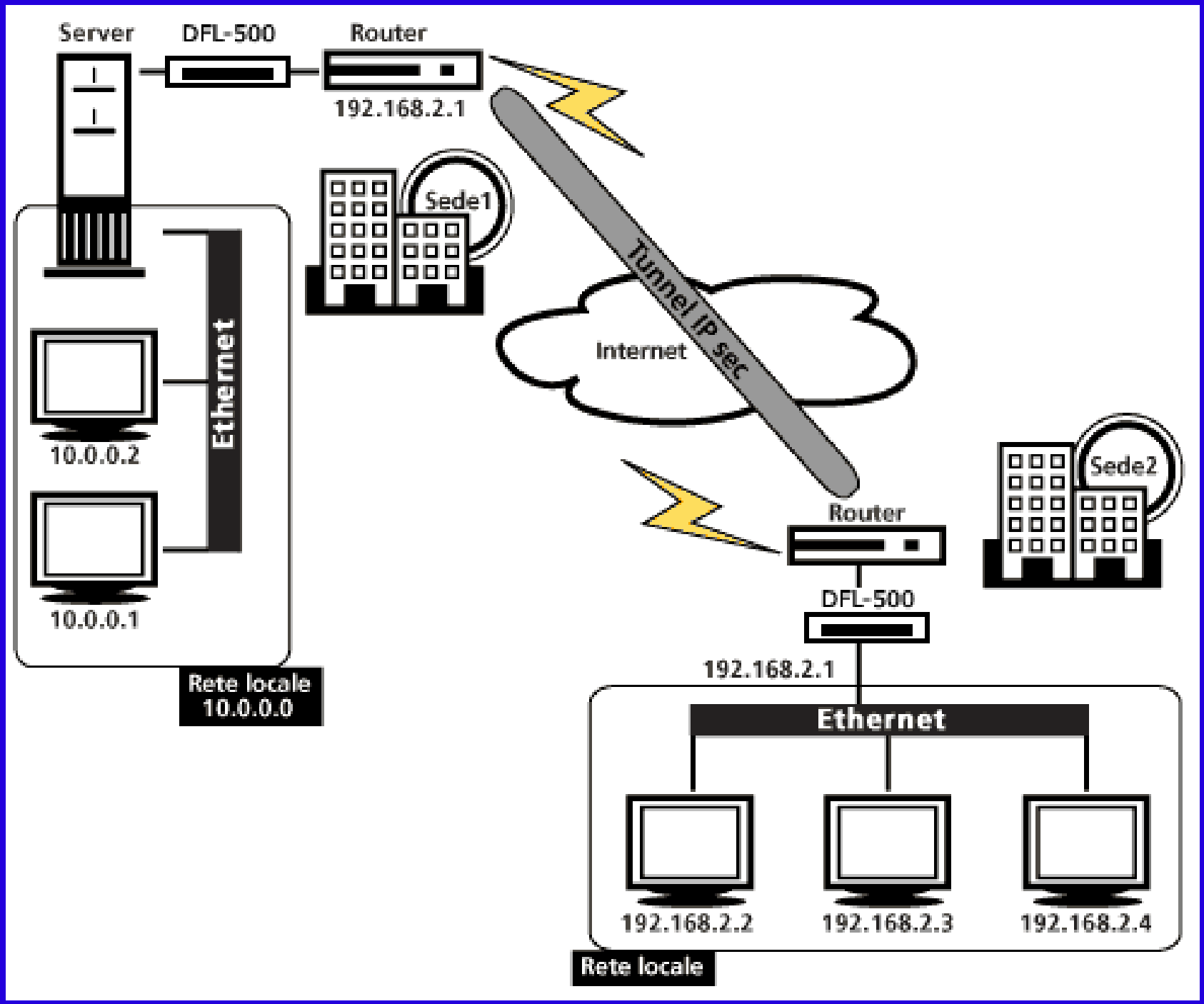
Router A

```
Ip tunnel add istanbul mode gre remote 212.252.11.5
     \
     local 144.18.110.40 ttl 255
ip addr add 192.168.10.1 dev istanbul
ip route add 192.168.1.0/24 dev istanbul
```

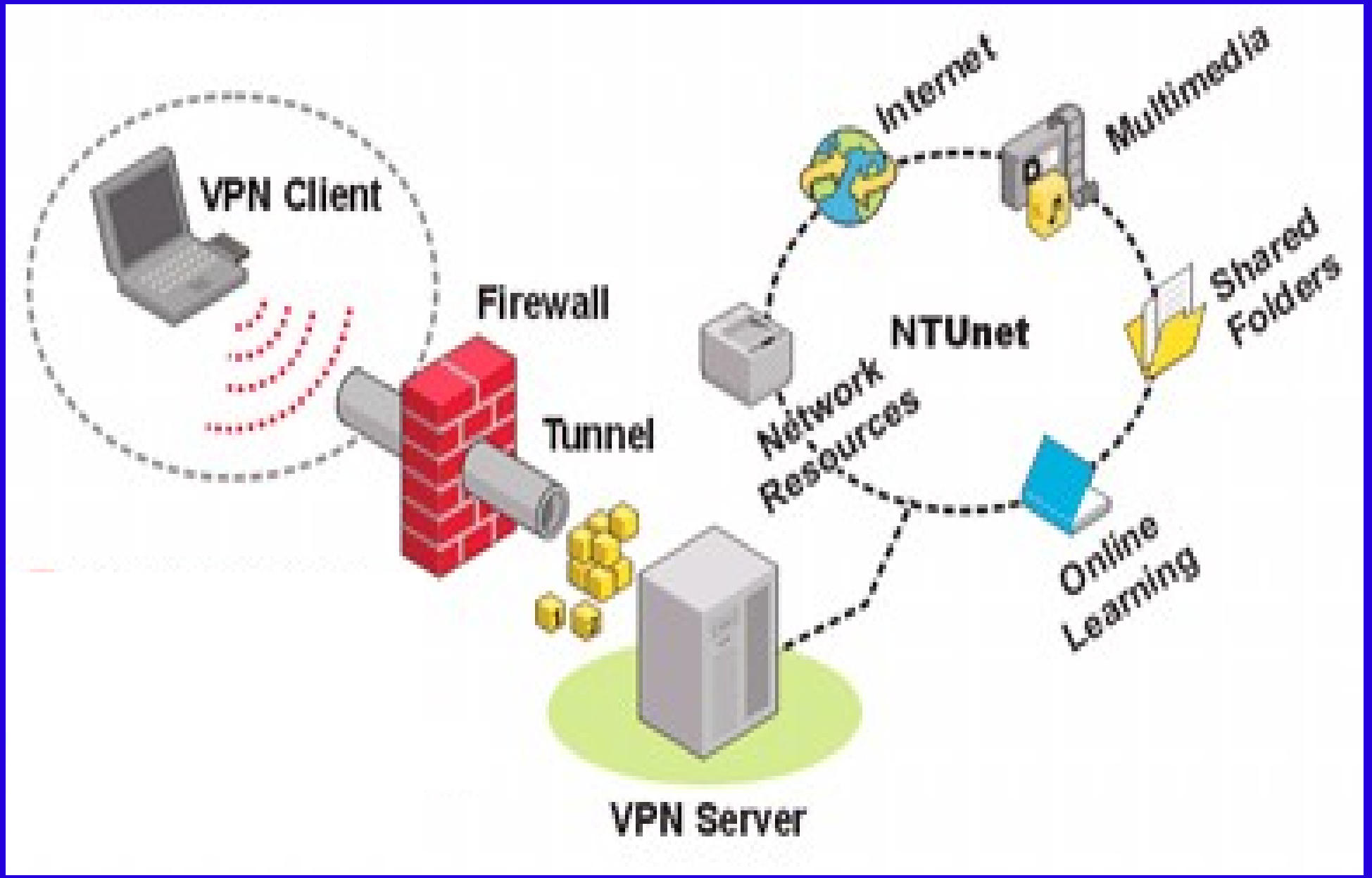
- Her iki uç birden internete bağlı ve IP adresleri biliniyor olmalıdır. Bu da ancak kiralık hatlarda elverişli olabilir.

Secure Tunnel - VPN

DES / DES3

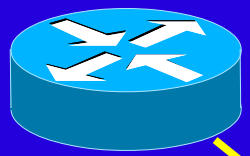


Secure Tunnel - VPN



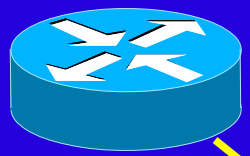
ssh, “secure shell” tanımından gelir. Telnet gibi benzer uzaktan bağlantı metotlarının ağ üzerindeki bilgileri herhangi bir şekilde gizlemeden taşıyor olmaları sorununa çözüm getirir.

ssh ile taşınan tüm veriler güçlü yöntemlerle şifrelenip performans kazancı için sıkıştırılmaktadır.



Kuşbakışı SSH

- SSH, istemci/sunucu mimarisi ile çalışır.
- İstemciler, sunucunun kendi istedikleri sunucu olduğunu kontrol edebilirler.
- İstemci ve sunucular için son derece detaylı erişim kontrolleri uygulanabilir.
- SSH, kendi kullanıcı doğrulama mekanizmasını sunar. Diğer yandan normal parola metotları da SSH ile kullanılabilir.
- SSH ile oluşturulan bir tünel diğerlerinden daha kolay ve esnek olabilirken yüksek düzey güvenlik sağlar.



VPN server sistemini mutlaka adanmış bir makine olarak hazırlayın.

Kapsamlı dağıtımlardan kaçının..

Mandrake, RedHat, SuSE vb.. dağıtımları tercih etmeyin.

Bilhassa router yapmak üzere hazırlanmış dağıtımlar..

SlackWare, DEBIAN gibi daha az paket kuran dağıtımlar..

Çalışan süreçleri en az sayıya indirin.

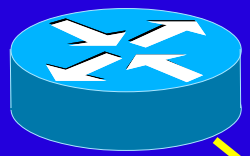
RPC, identd, httpd vs.. VPN için gerekmeyen süreçleri kapatın

Sistemde hiç bir kullanıcı olmamasını sağlayın.

Sisteme root dışında hiç bir “kullanıcı” eklemeyin.

Eklenecek kullanıcılar, sadece tünel bağlantısı için kullanılmalıdır.

Bu kullanıcıların da parolaları kapatılmalıdır.



VPN clientleri için gereken parolaları hazırlayın.

SSH Private key'ini oluşturun..

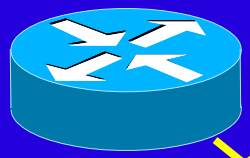
```
ssh-keygen -f /etc/ssh/ssh_host_key
```

Parolayı boş bırakın.

Böylece tünel bağlantısı otomatik olarak kurulabilir.

/etc/ssh/ssh_host_key* dosyalarının güvenliğini sağlayın.

Bu dosyaların sahibi mutlaka “root” olmalıdır.
Sadece sahibi tarafından okunabilir olmalıdır.



VPN clientleri için gereken parolaları hazırlayın.

VPN kullanıcıları için bir grup oluşturun.

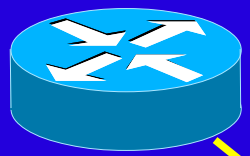
```
Groupadd vpn-users
```

VPN kullanıcıları için ev dizinini ve ssh özel dizinini oluşturun.

```
mkdir /home/vpn-users
mkdir /home/vpn-users/.ssh
chown -R root:vpn-users /home/vpn-users
```

VPN kullanıcılarını (yetkili sistemler) oluşturun.

```
useradd -d /home/vpn-users -g vpn-users sivas
    -s /usr/sbin/pppd
```



VPN clientleri için gereken parolaları hazırlayın.

VPN kullanıcıları için SSH Authentication Key'lerini oluşturun.

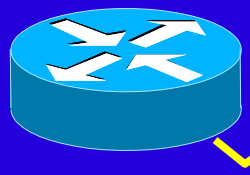
```
ssh-keygen -f /home/vpn-users/identity.vpn
```

Oluşan dosyanın içeriğini “authorized_keys” dosyasına ekleyin.

Oluşturulan key'dosyasını sadece disketle taşıyın.

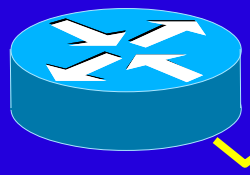
Bu dosyaların güvenlik açısından Ağ üzerinden iletilmesi sakıncalı olur.

Bu dosyalar, Tünel'in diğer ucunda kullanılacak olan parolaları ihtiva eder.



SSH sunucu yapılandırmasını düzenleyin /etc/ssh/sshd_config

PermitRootLogin yes
StrictModes yes
CheckMail no
X11Forwarding no
KeepAlive yes
RhostsRSAAuthentication no
PasswordAuthentication no
UseLogin no
IgnoreRhosts yes
QuietMode no
IdleTimeout 3d
PrintMotd no
RhostsAuthentication no
RSAAuthentication yes
PermitEmptyPasswords no



Yolbulma tablosunu Karşı ağa uyacak şekilde düzenleyin.

Varsayılan ağgeçidi

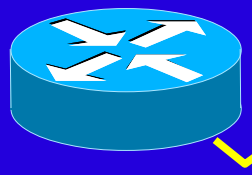
```
route add default gw a.b.c.d dev eth0
```

İnterneti ve doğal olarak tünelin öteki ucunu gösteren ağgeçidi.

Yerel ağınız için yolbulma bilgisi.

```
route add -net 192.168.0.1/24 dev eth0
```

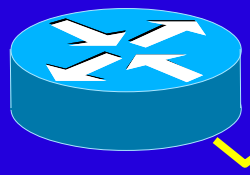
Yerel ağınızdaki makinelere erişim için gereken bilgi...



Host Trust Key'ini bu makineye yükleyin

Tüneli devreye alın

```
Pty-redir ssh -t -lsivas vpn_server \
  pppd noauth > /tmp/vpn-dev
sleep 10
$VPNDEV=`cat /tmp/vpn-dev`
pppd noauth $VPNDEV 10.1.1.1:10.1.1.2
sleep 5
route add 192.168.0.0 netmask 255.255.255.0 gw 10.1.1.1


ssh -t -lsivas vpn_server \
  route add 192.168.1.0 netmask 255.255.255.0 gw 10.1.1.2
```



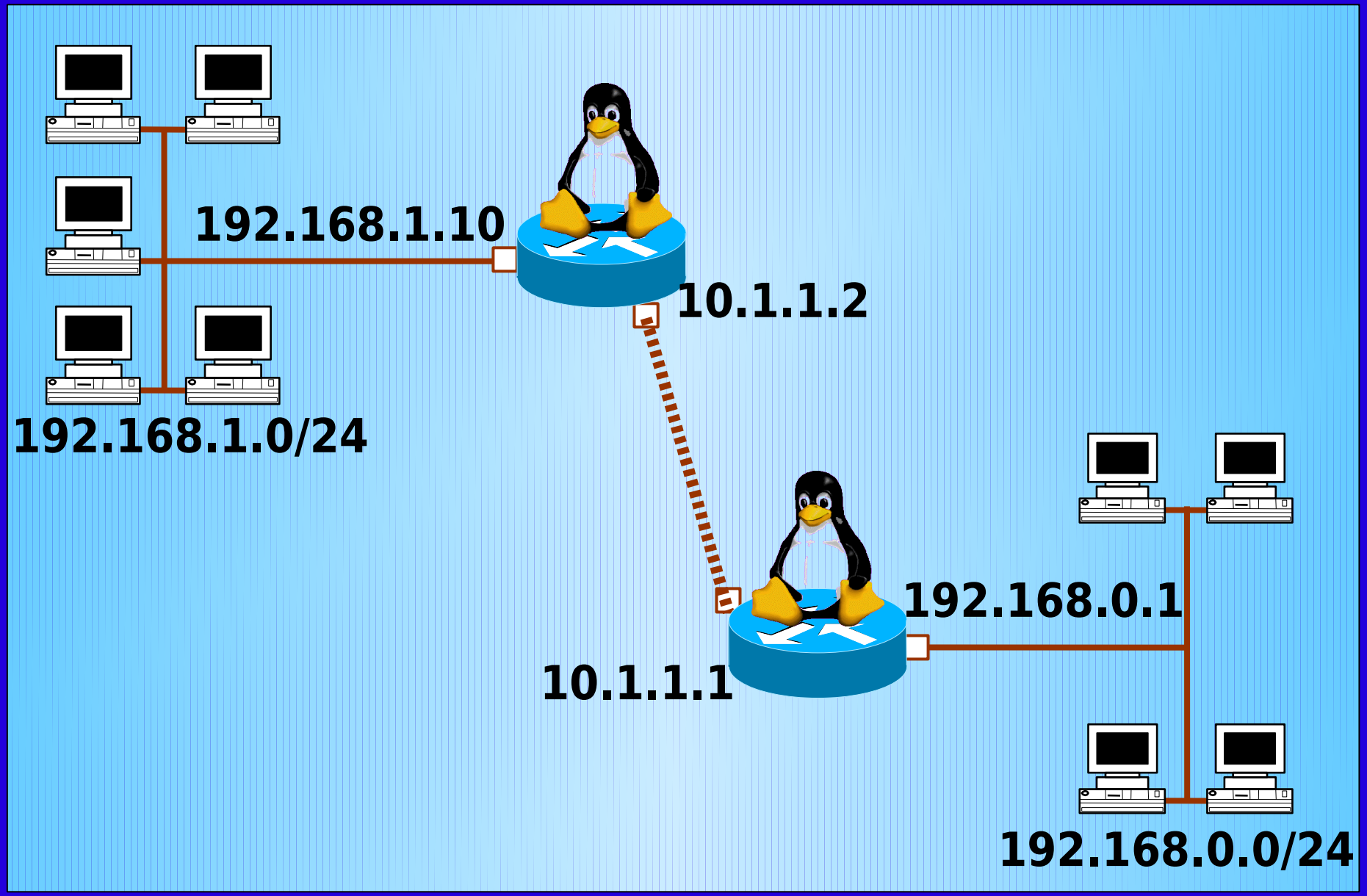
pty-redir ?

```
pppd updetach noauth passive pty "ssh -P vpn_server -l sivas
-o BatchMode=yes sudo pppd nodetach notty
noauth" ipparam vpn 10.1.1.1:10.1.1.2
```

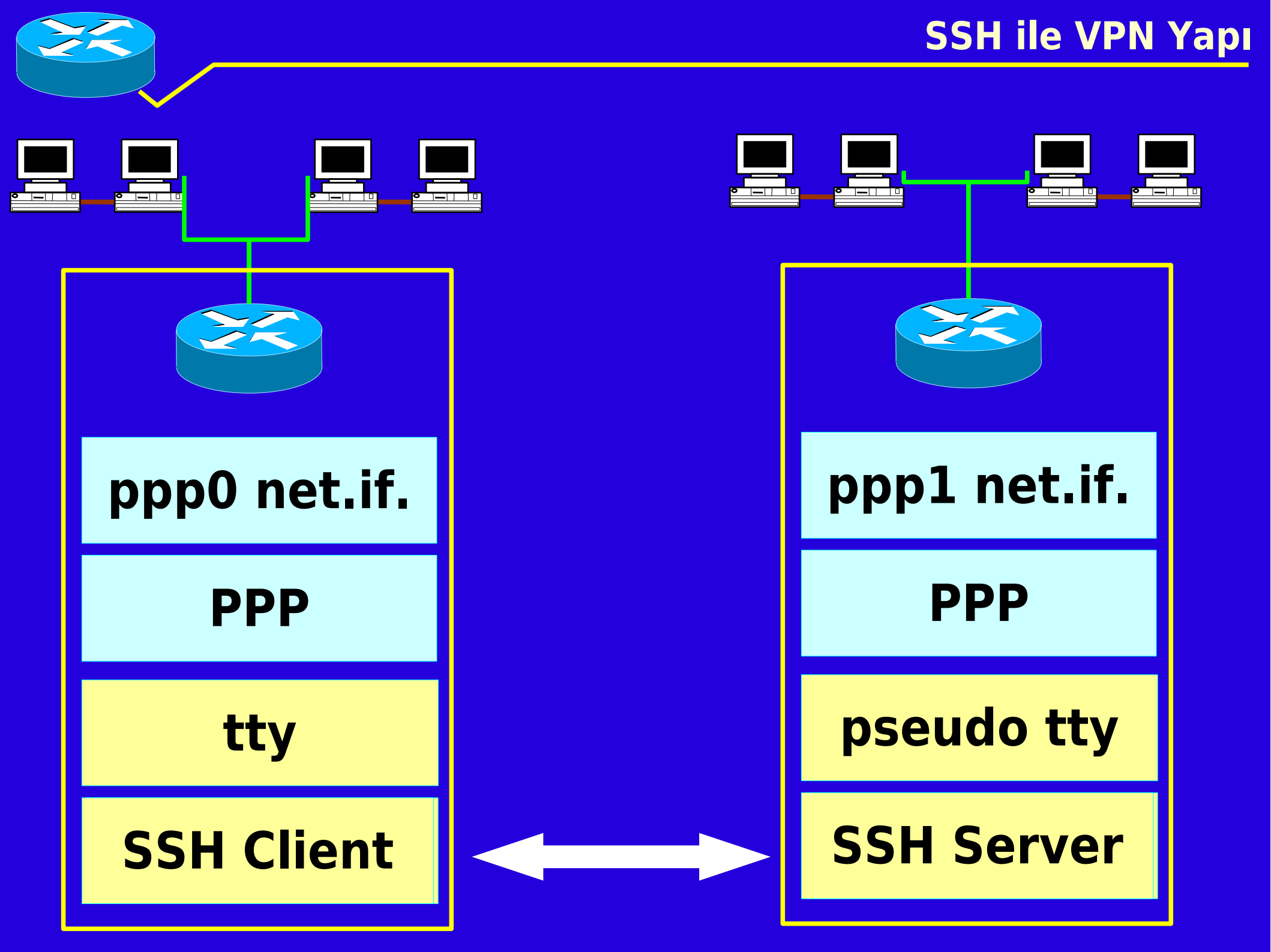
pty-redir, sadece 2.0 ve 2.2 kerneller için uygundur. 2.4 Serisi kernel ile birlikte kullanılan pppd 2.4, “pty” parametresi ile birlikte gerekli pseudo-tty'yi kendisi sağlayabilir.

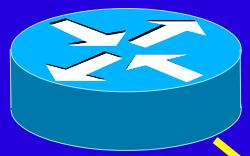
Bilhassa tek disketlik dağıtımların çoğu 2.2 serisi kernel kullanmaktadır.

SSH ile VPN – Ne Yaptık ?



SSH ile VPN Yapı





TCP Port Forwarding – En kolay Tünel..

SSH, kolayca yerel bir portu öteki tarafa taşıyabilir..

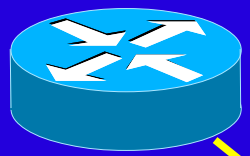
```
ssh -L 139:host:139
ssh -R 139:host:139
```

SSH, herhangi bir TCP portunu kolayca cliente taşıyabilir.

SSH, clientleri hemen hemen tüm O/S'lar için mevcuttur.

Gerek VPN Komple Tünel olarak, gerekse, port ilerletme özellikleri ile Windows, UN*X, OS/2, MacOS (X) için GPL ve Freeware olarak temin edilebilir.

Ya UDP, örneğin syslog çağrıları ?



UDP Port Forwarding

LOGSERVER üzerinde:

```
nc -l -p 9999 | nc localhost -u syslog
ssh -g -R 9999:localhost:9999 root@remoteSer
```

Log Kaydı Üreten sistemde:

```
nc -l -u -p syslog | nc localhost 9999
```

IPSec, IP paketlerini tamamen şifreli olarak taşıyan IP dışında bir protokoldür.

IPSec, Standart IP üzerine bindirilebilir.

Linux 2.5 Serisi Kernel, IPSec desteği ile geliştirilmektedir.

IPSec Tünel Uygulaması

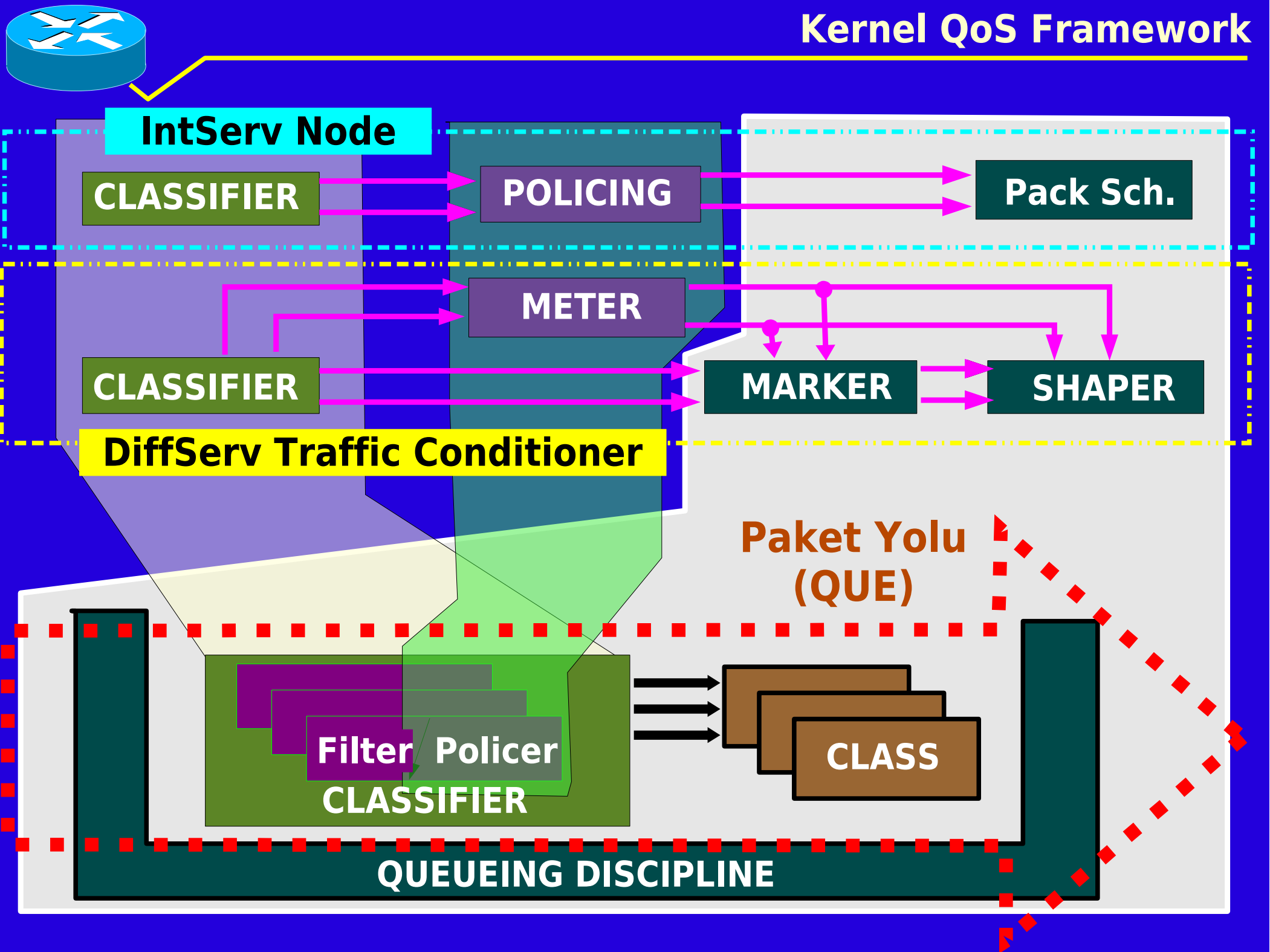
Full IPSec Firewall ve VPN Çözümüdür.

CheckPoint FW-1 VPN,
Cisco PIX VPN,
Microsoft W2K, XP (High Security Pack),

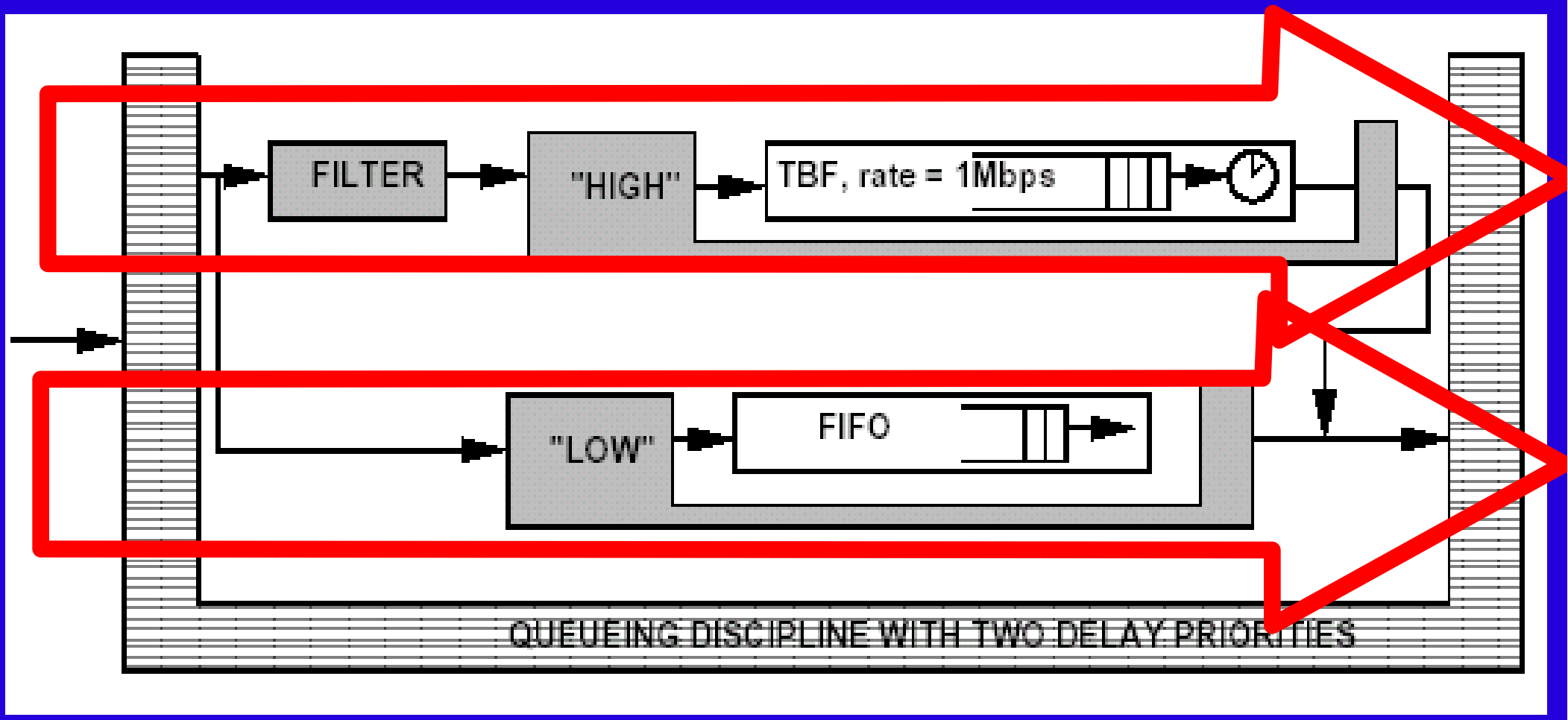
Gibi pek çok IPSec Sistemiyle uyumludur.

RoadWarrior uygulaması Dinamik IP ile mobil uygulamalara IPSec güvenliği ekleyebilir.

Kernel QoS Framework

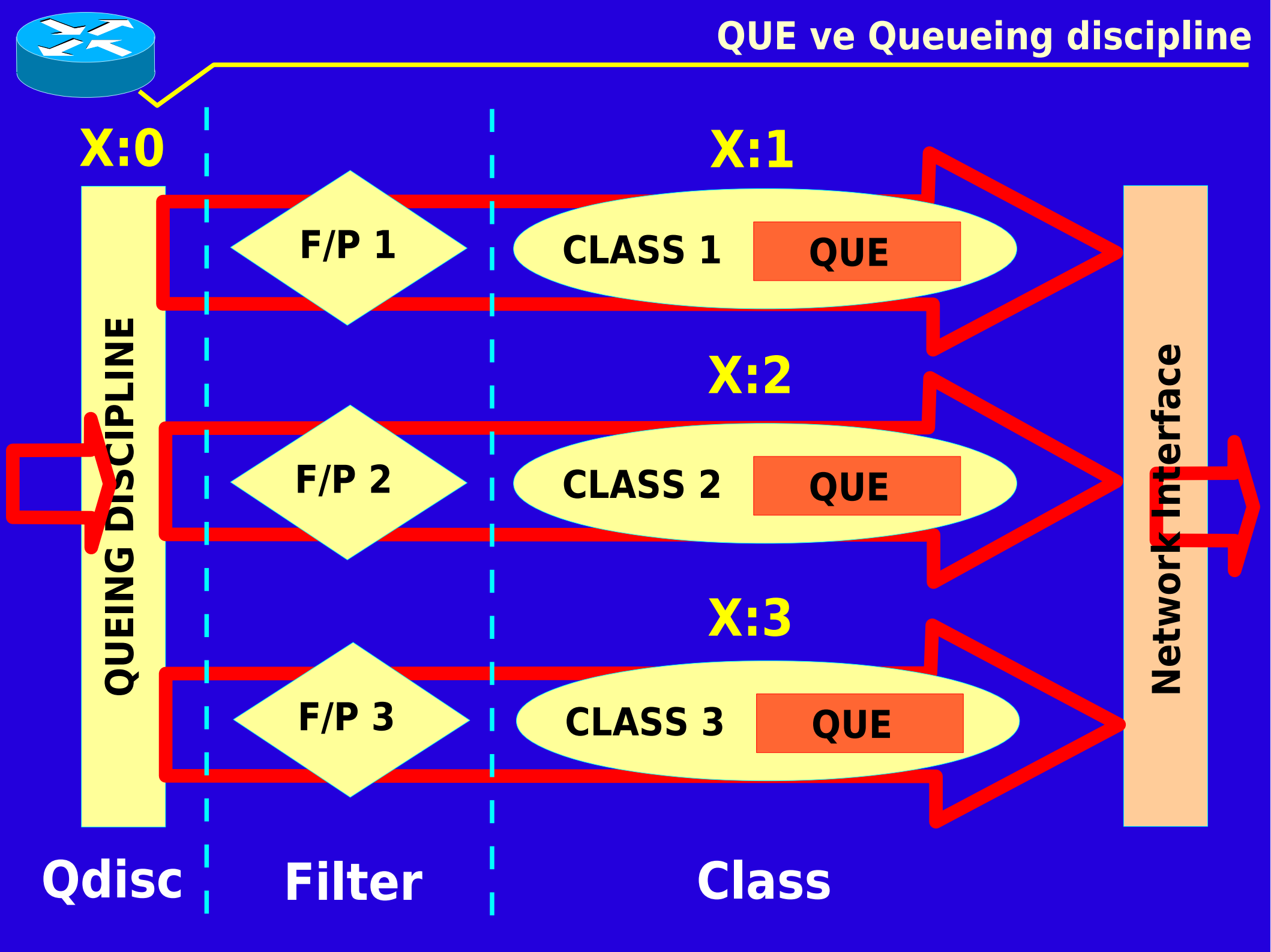


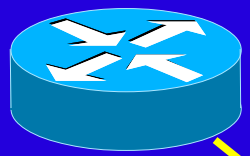
QUE ve Queueing discipline



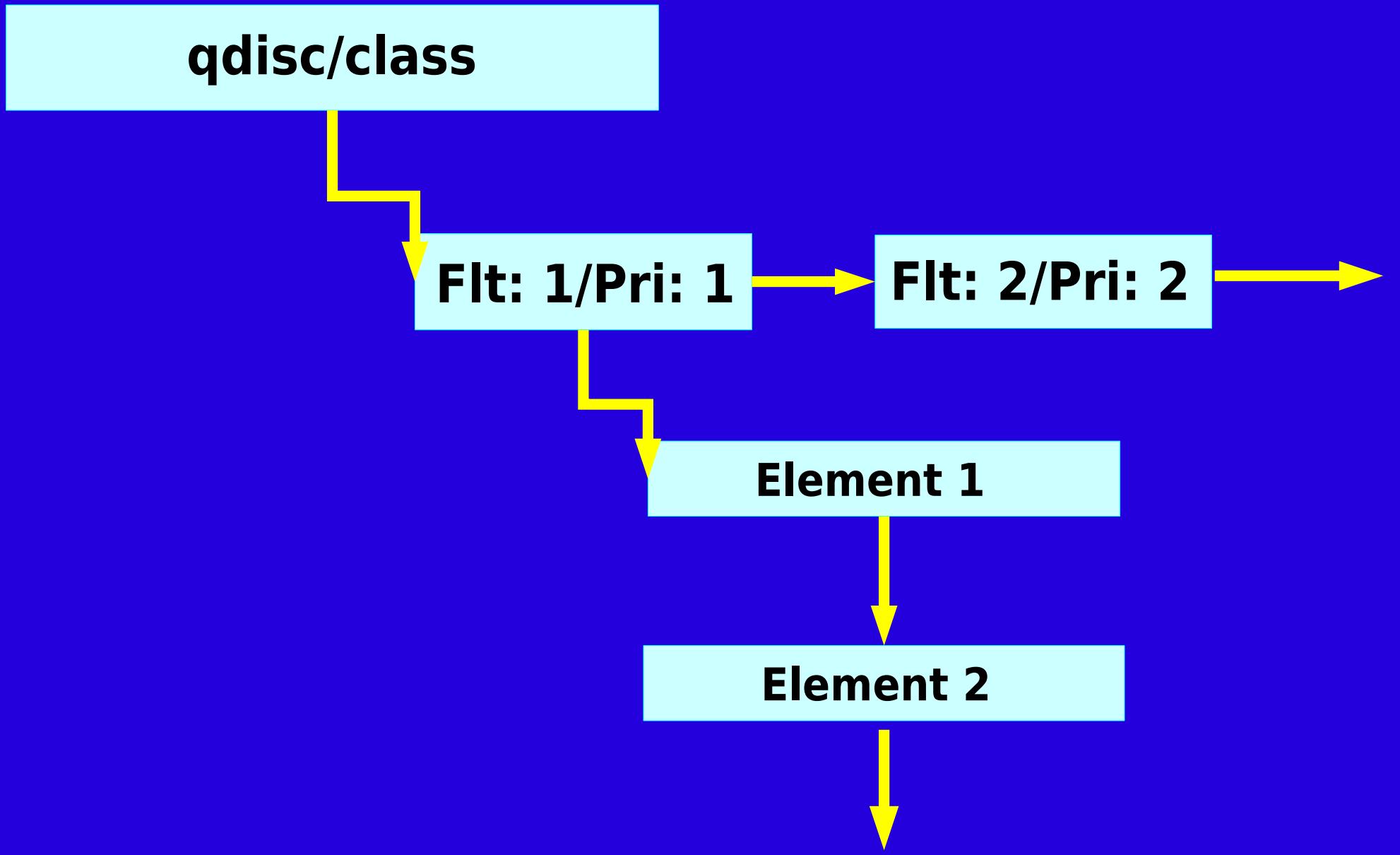
İki Gecikme/Bekleme öncelikli kuyruğa sahip bir kuyruk kuralseti

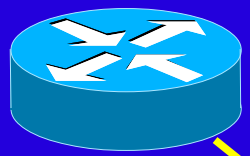
QUE ve Queueing discipline





Filter Priority

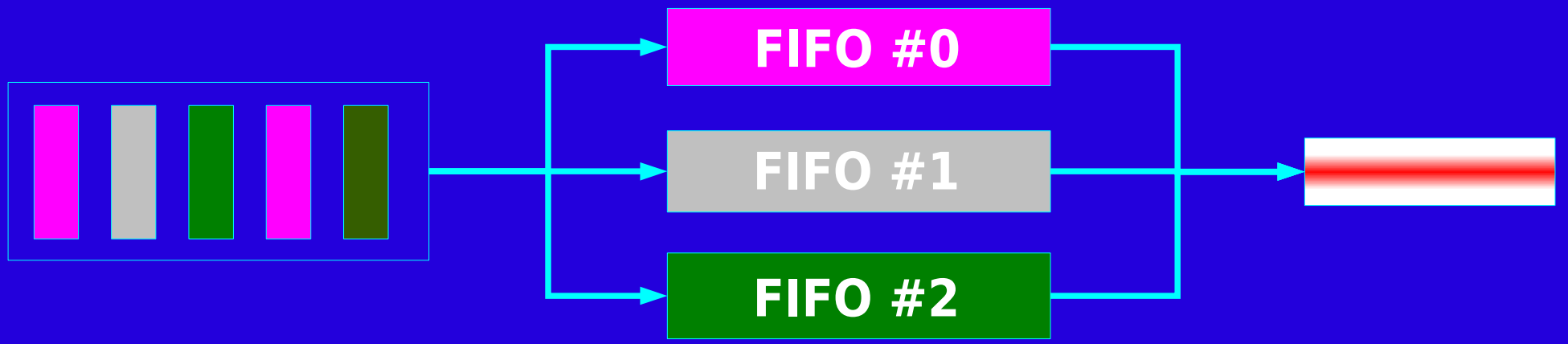




Kuyruk Kuralseti türleri

CBQ - Class Based Queueing
TBF - Tocken Bucket Flow
SFQ - Stochastic Fair Queueing..
FIFO - First In, First Out... (PFIFO_FAST)
TEQL - Traffic Equalizer..
HTB - Hierarchical Tocken Bucket..
PRIO- Priority
RED - Random Early Detection

FIFO (pfifo_fast) (Classless)



FIFO #0 - MD, MMC+MD, MMC+MD+MR, MD+MR

FIFO #1 - MT+MD, MMC+MT+MD, MR+MT+MD, MMC+MR+MT+MD, NS, MR, MMC+MR

FIFO #2 - MMC, MT, MMC+MT, MR+MT, MMC+MR+MT

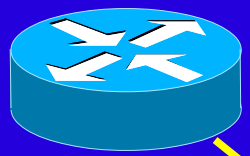
MMC - Minimize Monetary Cost

MT - Maximum Throughput

MR - Maximum Reliability

MD - Minimize Delay

NS - Normal Service



TBF Kuralseti (classless)

rate based

Kuyruğa girecek paketler basitçe orantı yoluyla belirlenir.

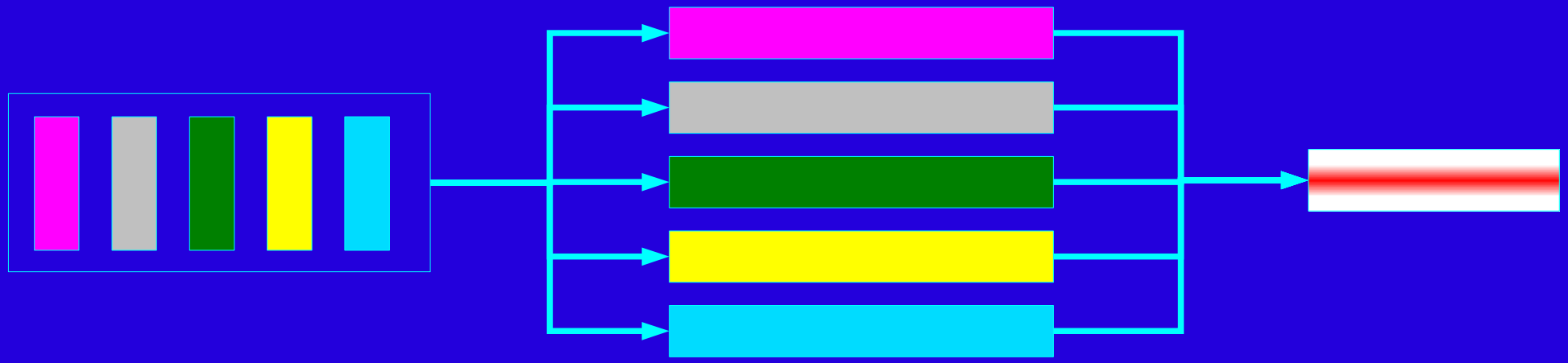
Basit, network ve CPU dostu.

Gerek arabellek yönetimi, gerekse ağa yollama açısından basit bir algoritma kullanır.

Interface bandwith'i düşürmek için en uygun qdisc

Sözde oturumların herbiri için ayrı bir FIFO kullanımı..

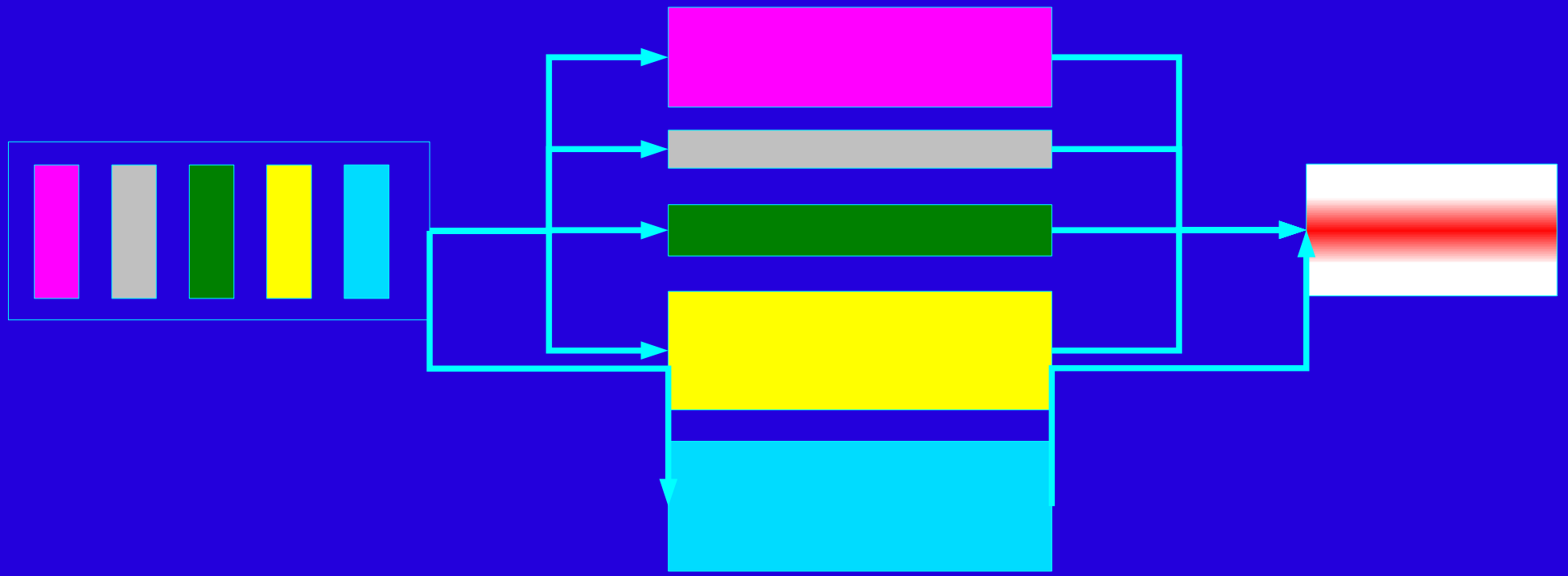
Router'a yapılan her oturum için ayrı birer FIFO oluşturup round-robin yoluyla paketleri bunlara dağıtır.

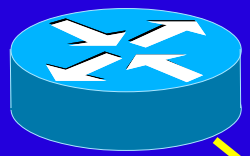


PRIO Kuralseti (Classfull)

Band tabanlı kuyruklar oluşturur.

- Her band ayrı önceliğe sahiptir.
- Bantlar FIFO olarak düzenlenir, istenirse her band farklı bir qdisc şeklinde hazırlanabilir.





Exponential Weighted Moving Average

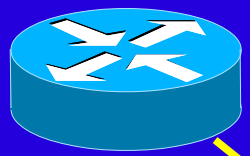
Kuyruktaki ortalama bekleme süresini çok doğru hesaplayabilir. Hesaplama paket bazındadır.

Borrowed & Bounded priority

Yüksek öncelikli kuyruklar boşsa, düşük öncelikli olanlar bantgenişliğini ödünç alabilir.

BURST modunda çalışabilme

Ardı ardına paketleri hızla ve kesintisiz yollayabilir.



CBQ Parametreleri

avpkt - Ortalama paket boyu.

bandwith - Arabirimin bant genişliği.

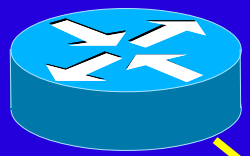
cell - Arabirimden çıkacak paket için bölme faktörü

maxburst - Burst modunda max paket sayısı.

minburst - Görev yürütme sürecinde gönderilecek en az paket sayısı

minidle - avgidle için minimum negatif değer.

mpu - Minimum paket boyu, arabirim için..



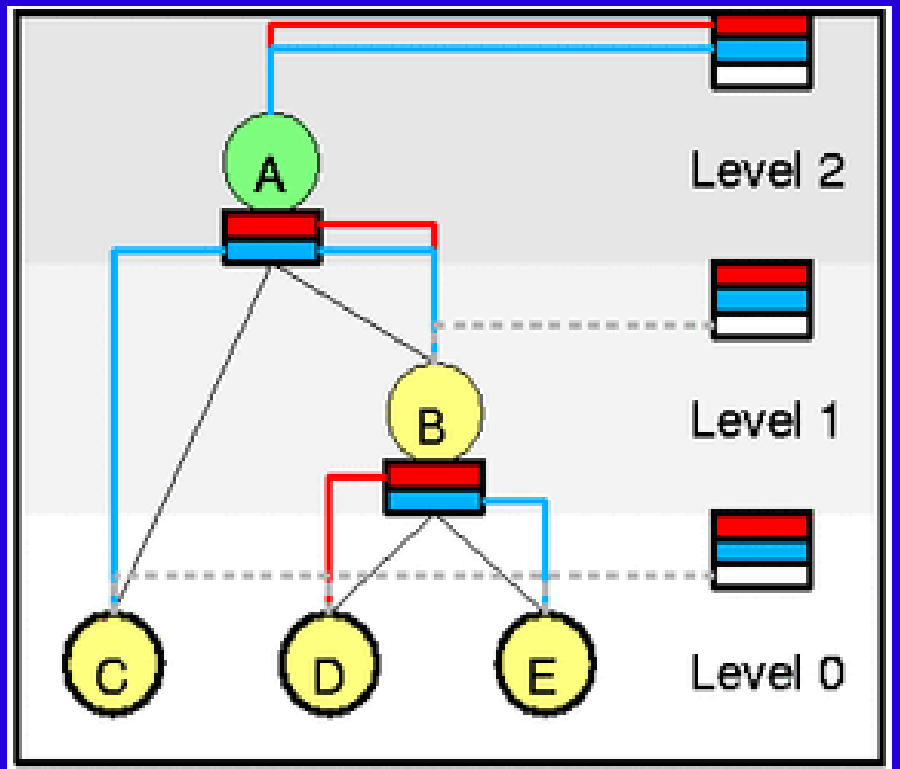
CBQ Parametreleri

rate	-	CBQ için farzedilen bantgenişliği.
allot	-	Arabirim paket boyu.
weight	-	Boş olduğu farzedilen bant aralığı.
isolated		
shared		Diğer que'ler boş bandı kullanabilir
bounded		
borrow		Diğer que'lerdeki boş bandı kullan

Classful TBF implementasyonu.

Her band ayrı önceliğe sahiptir.

Bantlar FIFO olarak düzenlenir, istenirse her band farklı bir qdisc şeklinde hazırlanabilir.

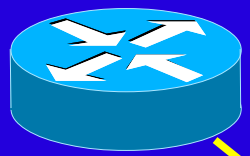


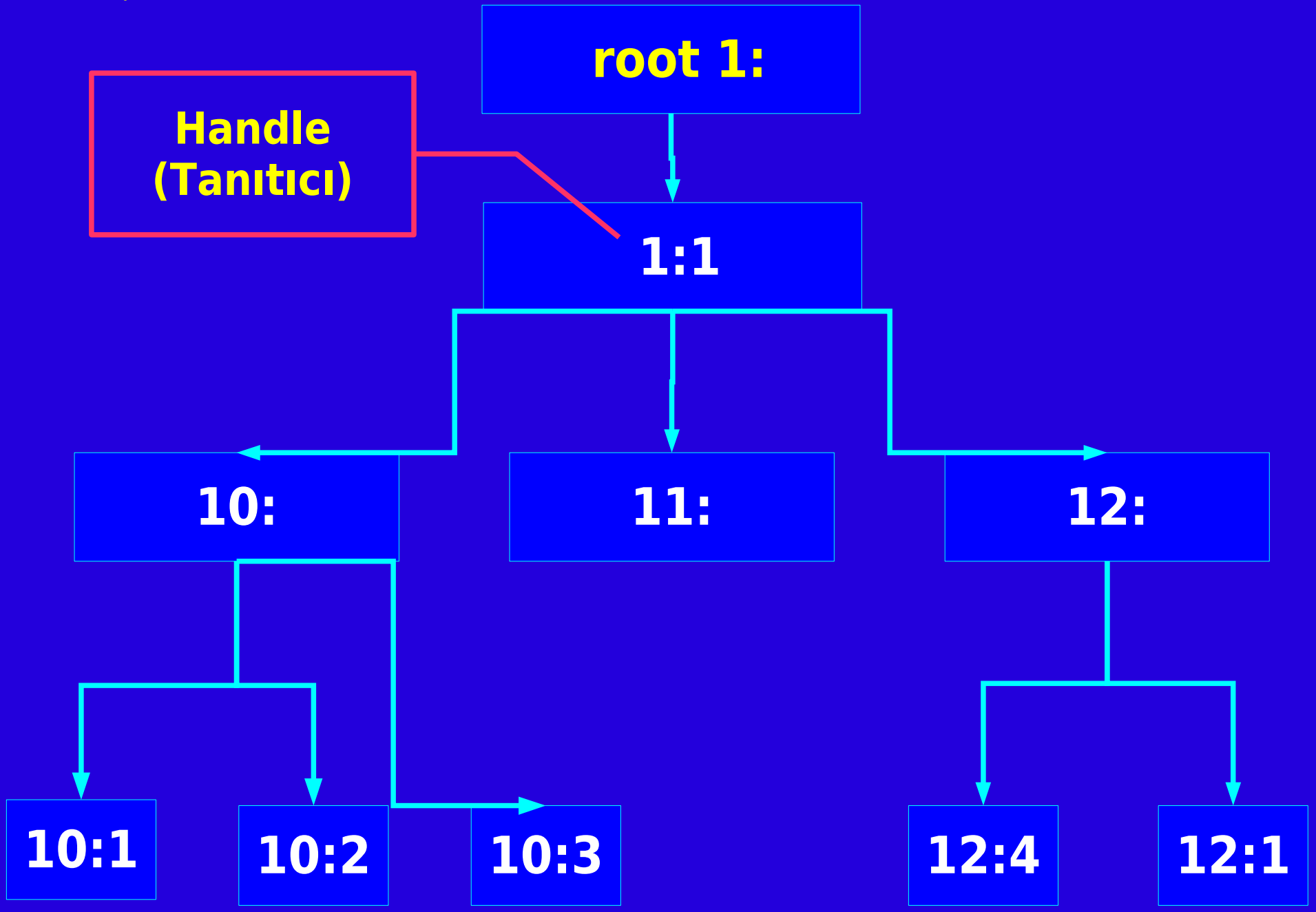
6. A can send, E and C can borrow

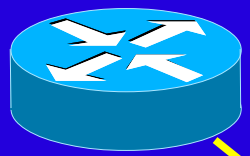
Classful QUE için CBQ, tipik bazı durumlarda yeterli performansı veremeyebilir.

HTB Bu durumlarda daha iyi netice verecektir.

Class Hierarchy







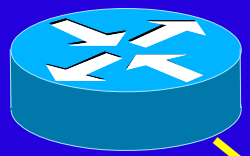
Sadece giden paketler için QoS/TC Yapılabilir.

Gelen paketler karşı tarafın insiyatifindedir. Aksi olsa çok güzel olur, DDoS denen kavram hiç olmazdı :)

Eğer paketlerin ulaştığı router, kendi paket yönetimini uygularsa, yapılan tanımlar geçersiz olur.

Paketler bir sonraki routerin kuyruğunda şekillendirilmemelidir. Bu, pratikte karşı routera sadece bize ayırdığı bantgenişliği kadar hızda erişerek sağlanabilir.

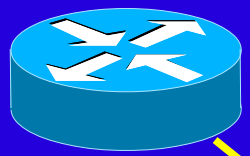
Giden paketlerin görünümü çok iyi ayarlanıp, gelen paketlerin giden paketlerin uyduğu kurallara uyması sağlanabilir.



Pratik TC örnekleri

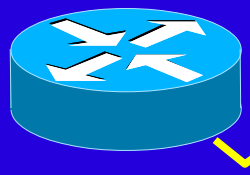
```
tc qdisc [ add | del | replace | change | get ] dev STRING
        [ handle QHANDLE ] [ root | parent CLASSID ]
        [ estimator INTERVAL TIME_CONSTANT ]
        [ [ QDISC_KIND ] [ help | OPTIONS ] ]
```

dev - qdisc'in bağlanacağı cihaz..
handle - qdisc'i tanıtıcı numara (parent : self)
root - qdisc'i köke bağla..
parent - qdisc'in hangi class'a bağlanacağı (tanıtıcı).
estimator - qdisc için zamanlama parametreleri.
qdisc-kind - qdisc'in türü, CBQ, TBF, FIFO, HTB vs...




```
tc qdisc add dev eth0 root handle 1:cbq \
   bandwidth 10M bitcell 8 avpkt 1000 mpu 64
```

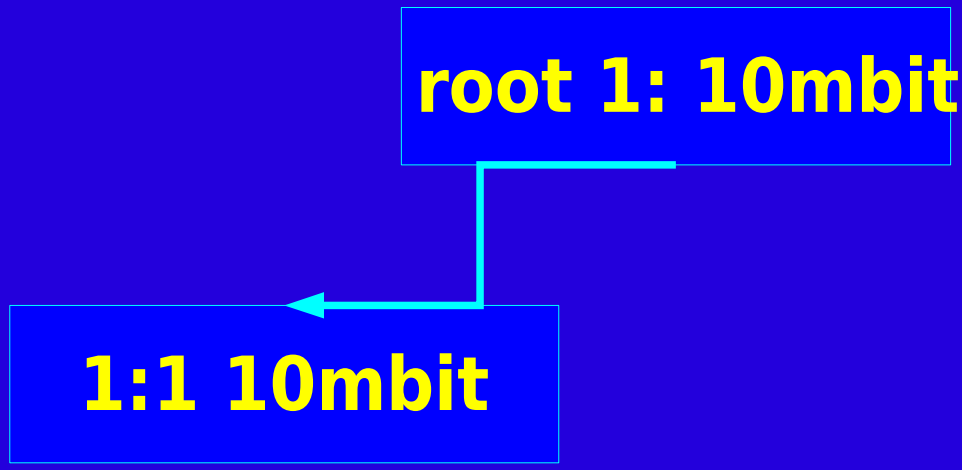
root 1:

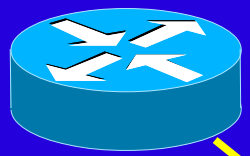



```
tc class [ add | del | change | get ] dev STRING
    [ classid CLASSID ] [ root | parent CLASSID ]
    [ [ QDISC_KIND ] [ help | OPTIONS ] ]
```

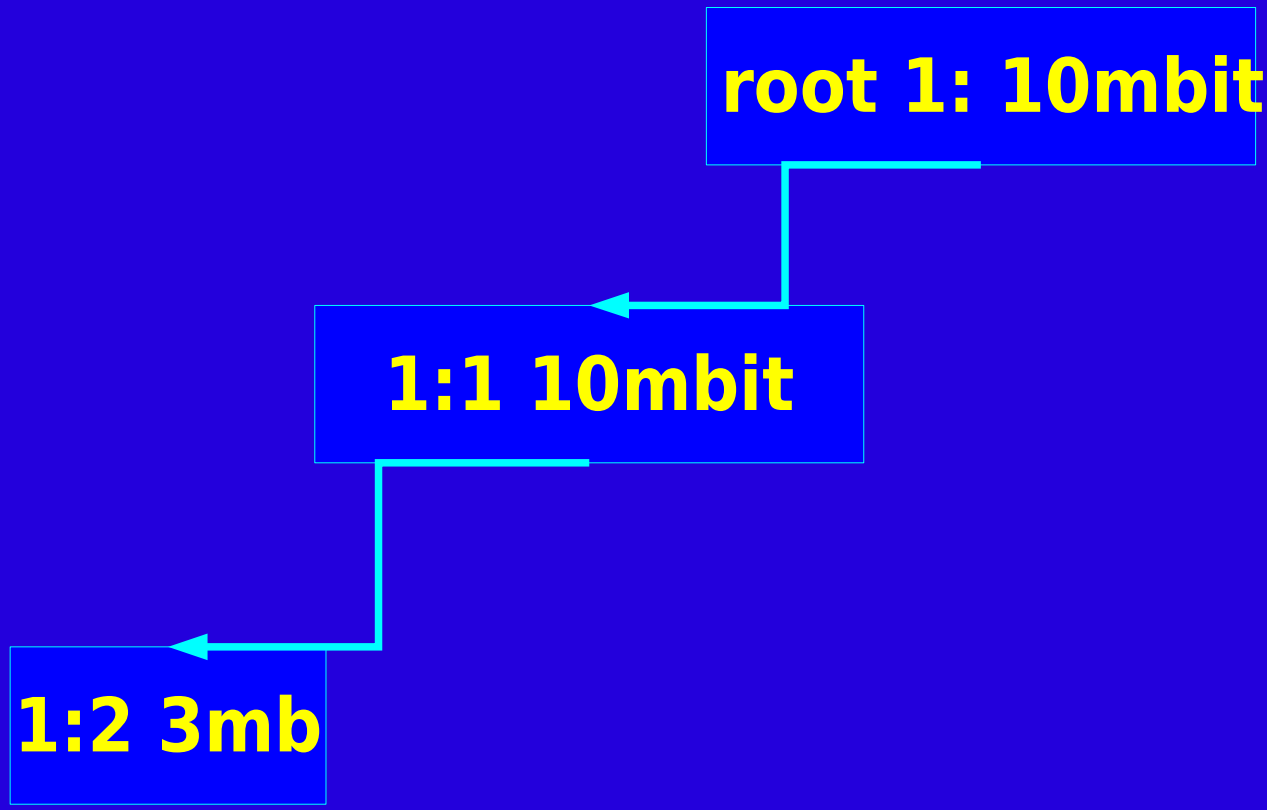
dev - class'ın bağlanacağı cihaz..
classid - class'ı tanıtıcı numara (parent : self)
root - class'ı köke bağla..
parent - class'ın hangi qdisc/class'a (tanıtıcı) bağlanacağı.
qdisc-kind - class'ın türü, CBQ, TBF, FIFO, HTB vs...

```
tc class add dev eth0 parent 1:0 \
   classid 1:1 cbq bandwidth 10M bit \
   rate 10M bit allot 1514 cell 8 \
   weight 1M bit prio 8 m axburst 20 avpkt 1000
```

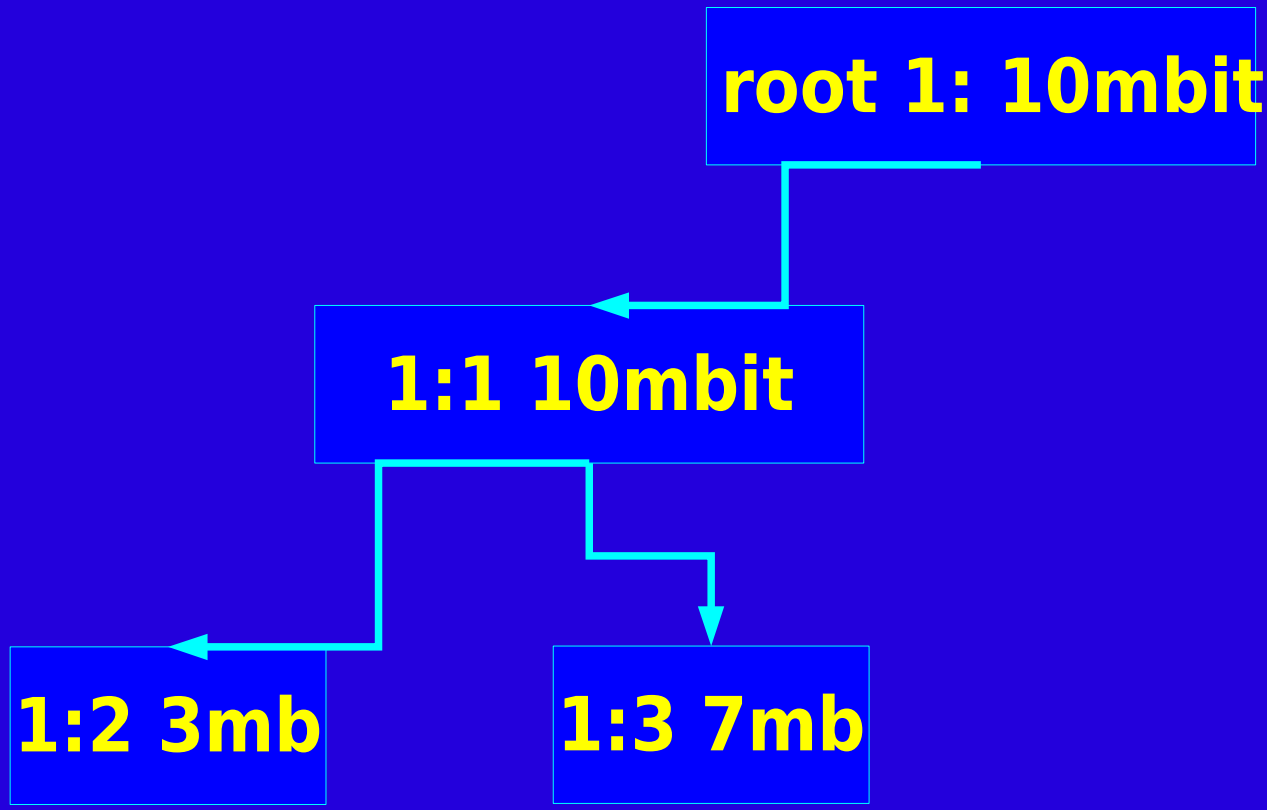




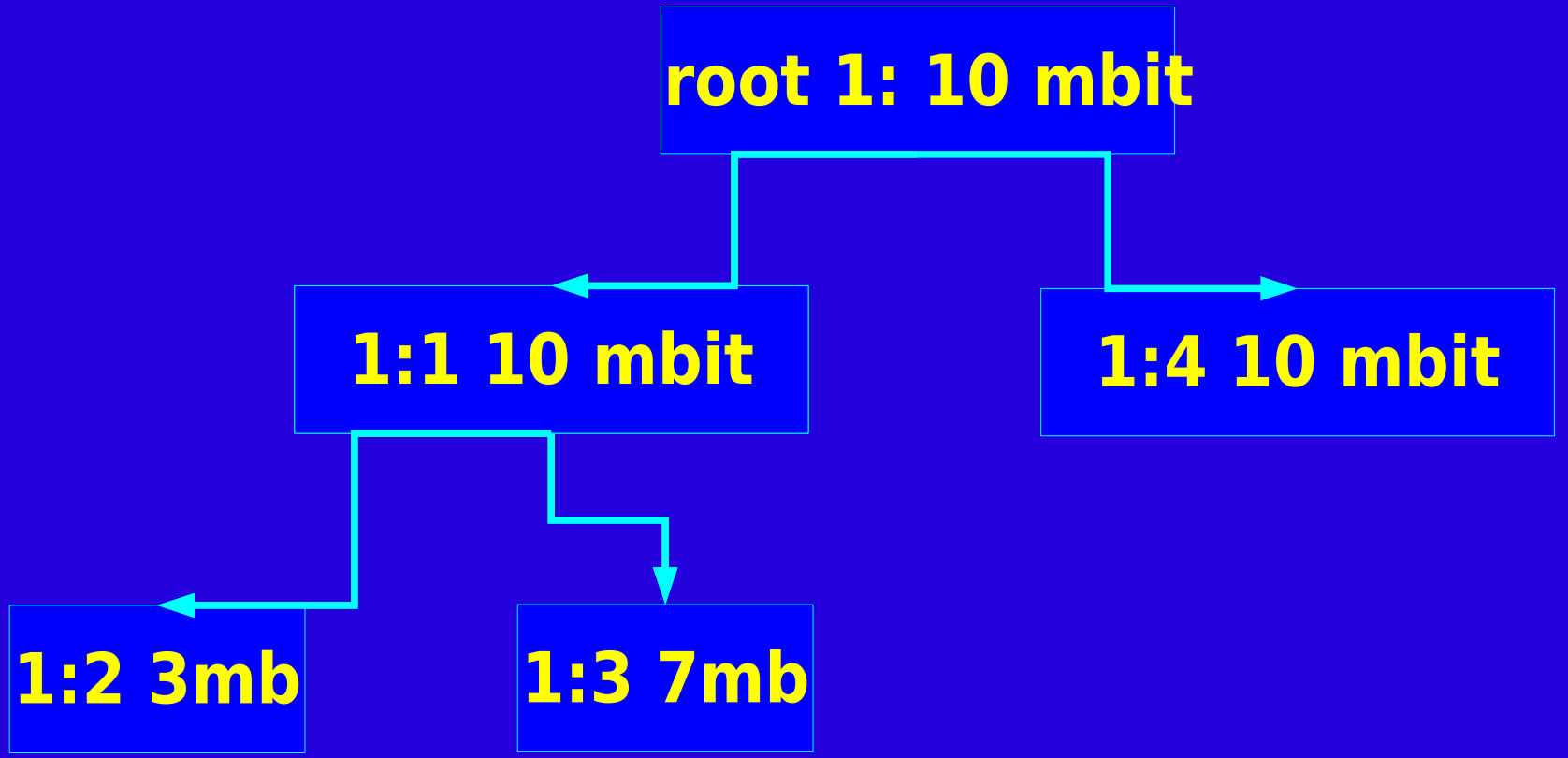
```
tc class add dev eth0 parent 1:1 \
   classid 1:2 cbq bandwidth 10Mbit \
   rate 3Mbit allot 1514 cell 8 \
   prio 3 maxburst 20 avpkt 1000 bounded
```



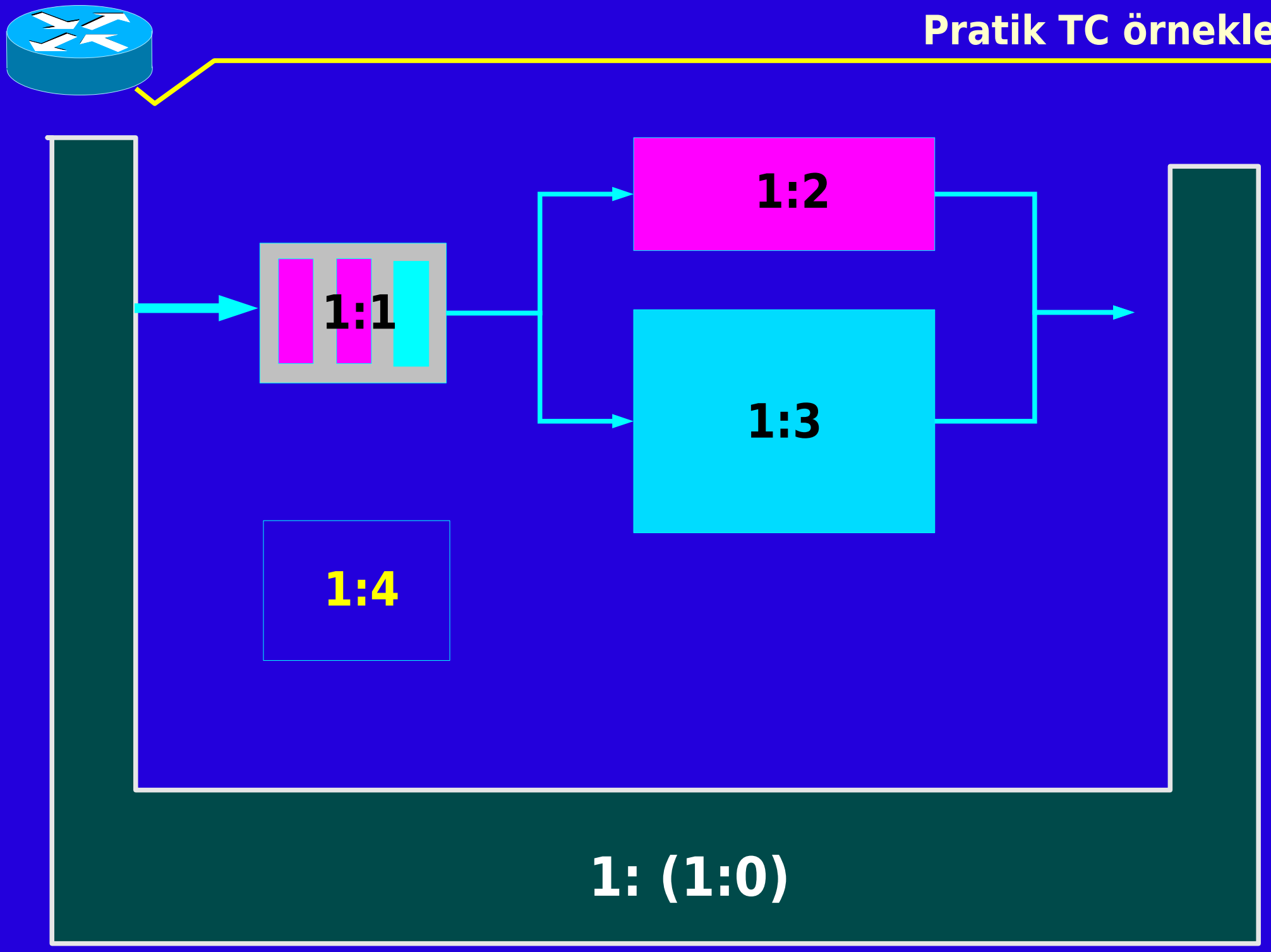
```
tc class add dev eth0 parent 1:1 \
   classid 1:3 cbq bandwidth 10Mbit \
   rate 7Mbit allot 1514 cell 8 \
   prio 7 maxburst 20 avpkt 1000 bounded
```

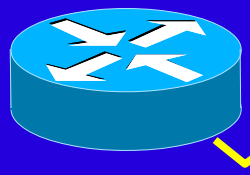


```
tc class add dev eth0 parent 1:0 \
   classid 1:4 cbq bandwidth 10M bit \
   rate 10M bit allot 1514 cell 8 \
   weight 1M bit prio 8 m axburst 20 avpkt 1000
```



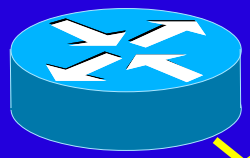
Pratik TC örnekleri

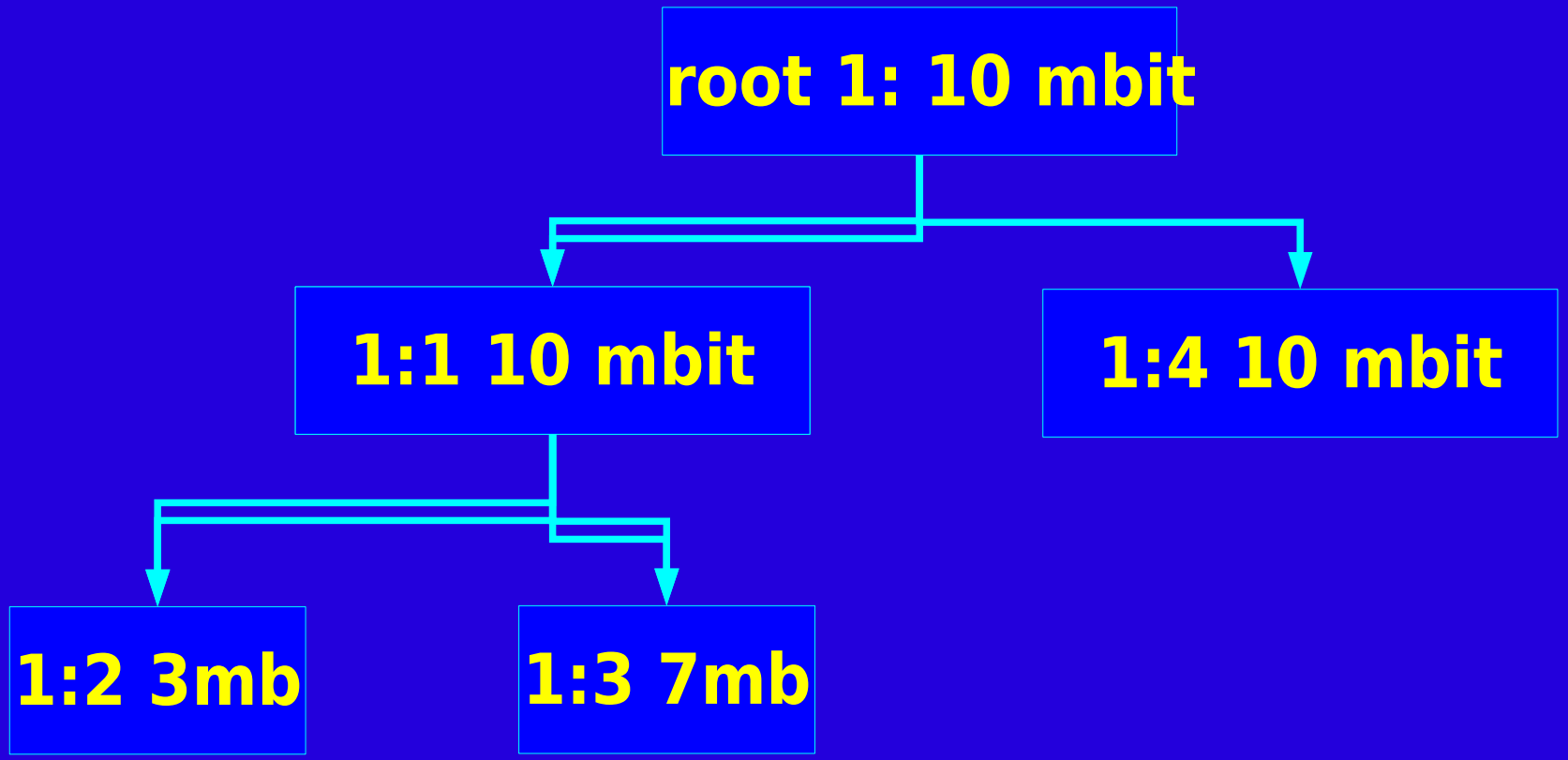




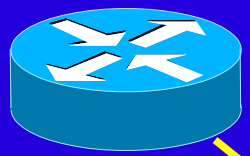

```
tc filter [ add | del | change | get ] dev STRING
        [ protocol PROTO ]   [ handle FILTERID ]
        [ root | parent CLASSID | classid CLASSID ]
        [ [ FILTER_TYPE ] [ help | OPTIONS ] ]
```

dev - filter'in bağlanacağı cihaz..
handle - filter'ı tanıtıcı numara (parent : self)
root - filter'i köke bağla..
parent - filter'in hangi class'a bağlanacağı (redirect).
classid,flowid - filter'in hangi class'a bağlanacağı (accept).
filter_type - Filtre türü, fw, route, u32 vs...






```
tc filter add dev eth0 proto ip parent 1:1 \
   prio 1 u32 match ip sport 80 flowid 1:2

tc filter add dev eth0 proto ip parent 1:1 \
   prio 2 u32 match ip dport 25 classid 1:3
```



Filter - Classifier Türleri

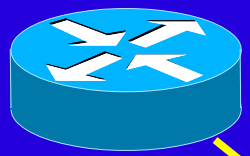
u32 - Packet üzerindeki Field'ler (SrcIP, DstIP vs.)

fw - netfilter tarafından işaretlenmiş paketler.

route - Yolbulma sürecinde belirlenen yol..

rsvp - RSVP Tariflerine göre (Sadece LAN için)

tcindex - DiffServ DSMARK işaretleri.



U32 Classifier

```
u32   [ match SELECTOR ... ] [ police POLICE_SPEC ]
      [ < classid |  flowid > CLASSID ]
```

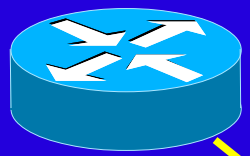
tcp src kaynak-port
dst hedef-port

udp src kaynak-port
dst hedef-port

ip protocol ip-protokolu
sport kaynak-port
dport hedef-port
src kaynak-adresi
dst hedef-adresi

{ u8 | u16 | u32 } değer maske **at** ofset

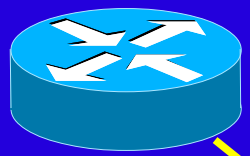
`match u8 128 0xFF at 8` **(Offset 8: TTL)**



ROUTE Classifier

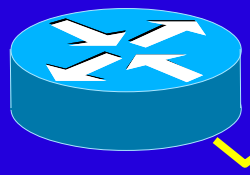
```
route     [ from REALM | fromif TAG ] [ to REALM ]
          [ flowid  CLASSID ]
```

```
ip route add dest via gw  dev device realm  id

ip route add src dev device realm  id
```



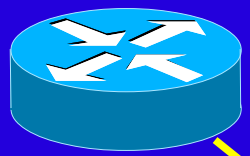
FW Classifier

```
iptables -t mangle -A PREROUTING \
      -s a.b.c.d -p tcp --dport 25 \
      -j MARK --set-mark 4

tc filter add dev eth1 protocol ip \
      parent 1:1 handle 4 fw flowid 1:3
```

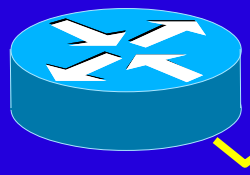


Örnek işlevler

```
tc qdisc add dev $DEV root handle 1:cbq \
    avpkt 1000 bandwidth 10mbit

tc class add dev $DEV parent 1:classid 1:1 cbq
    rate 512kbit allot 1500 prio 5 \
    bounded isolated

tc filter add dev $DEV parent 1:\
    protocol ip prio 16 u32 \
    match ip dst 195.96.96.97 flowid 1:1
```

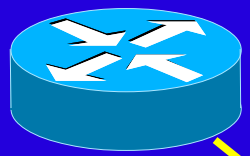



```
tc qdisc add dev $DEV root handle 1:cbq \
    avpkt 1000 bandwidth 10mbit

tc class add dev $DEV parent 1:classid 1:1 cbq
    rate 512kbit allot 1500 prio 5 \
    bounded isolated

tc filter add dev $DEV parent 1:\
    protocol ip prio 16 route

  ip route add 192.168.1.0/26 dev $DEV flow 1:1
```



Örnek İşlevler

```
tc qdisc add dev $DEV root handle 1:cbq \
    avpkt 1000 bandwidth 10mbit

tc class add dev $DEV parent 1:classid 1:1 cbq
    rate 512kbit allot 1500 prio 5 \
    bounded isolated

tc filter add dev $DEV parent 1:\
    protocol ip prio 16 route from  5 flowid 1:1

 ip route add 192.168.1.0/26 dev $DEV realm 5
```



Diğer Kabiliyetler

Internet Speed Booster.

MTU, TOS Gibi değerler üzerinde ince ayarlar.

Load Balancer

WRRP Implementasyonu:

http://wipl-wrr.dkik.dk/wrr/

High Availiable Router:

VRRP Implementasyonu:

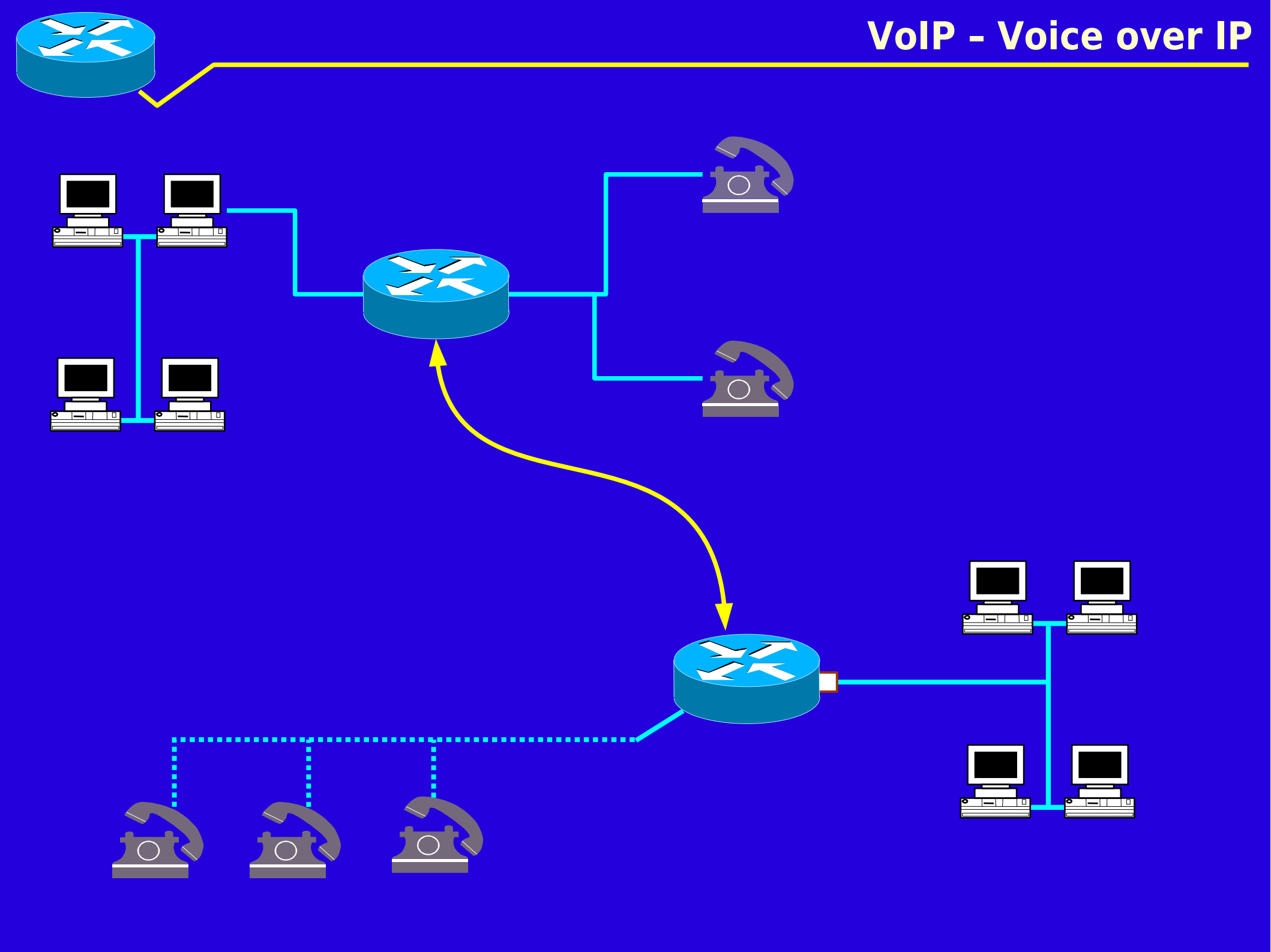
http://scry.wanfear.com/~greear/vlan.html

Packet Filter, DoS engelleme, Bridge Mode, TEQL ....

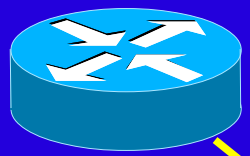
Temel firewall görevleri de dahil olmak üzere pek çok işlem sadece routing ile halledilebilir.

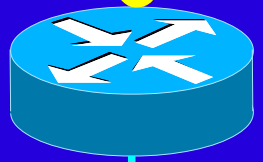
Daha pek çok routing ihtiyacı, arabirimler arası yük dengeleme, VLAN, OSPF/BGP/RIP vs. hepsi için yeterli destek mevcuttur.

VoIP - Voice over IP



VoIP Yapı





UDP/IP

RTP

Buffering

Compress

DSP

H323, H261

G723, G711 vs.

Echo Can. vs.

UDP/IP

RTP

Streaming

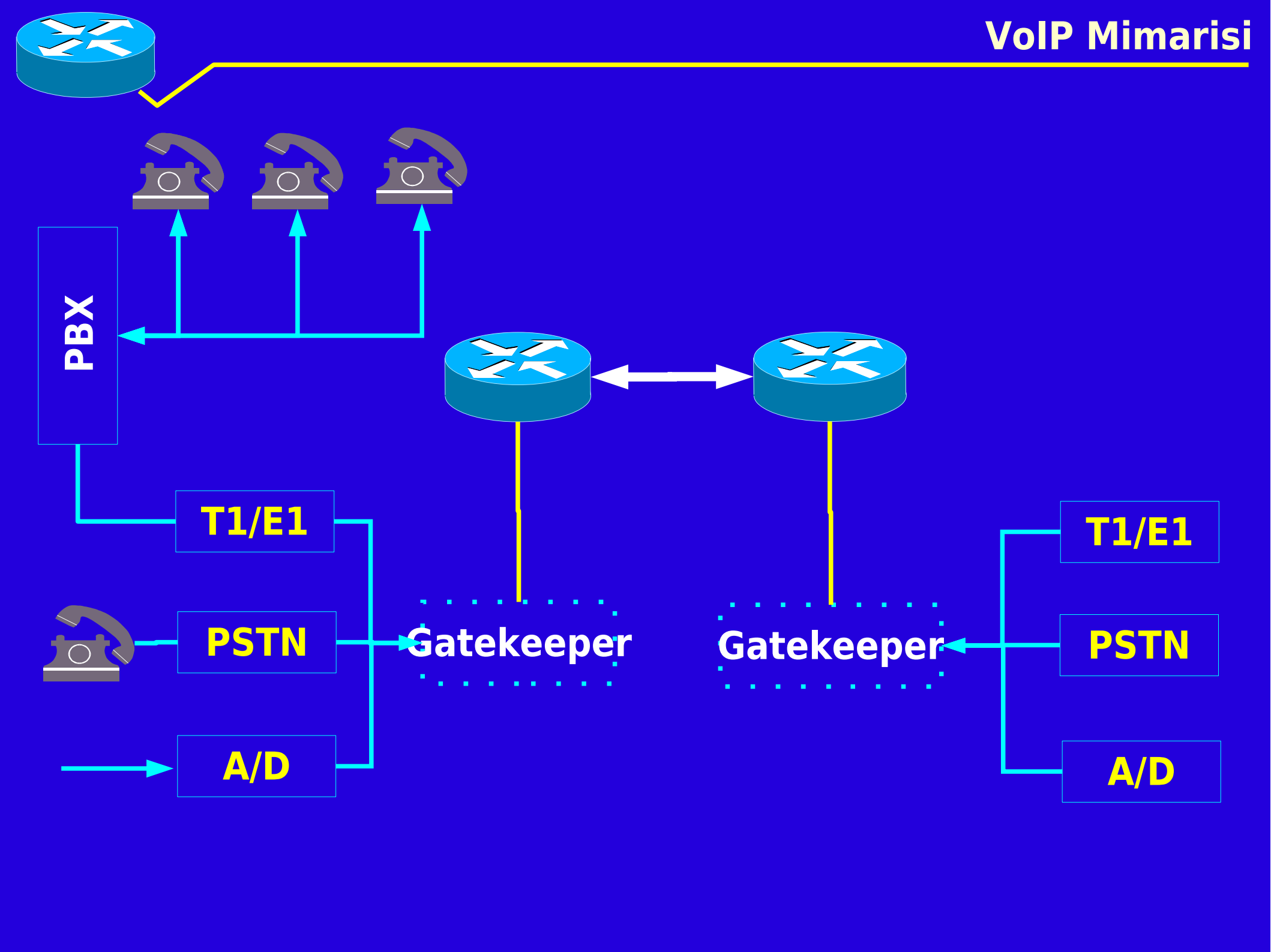
Decompress

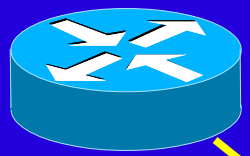
DSP

ADC

DAC

VoIP Mimarisi





T1/E1 - Sangoma, EiconCard vs. Standart Modemler..
PSTN - QuickJack, LineJack, VoicePump, VoiceTronix
A/D - FullDuplex Ses Kartları..

Hardware Avantajları

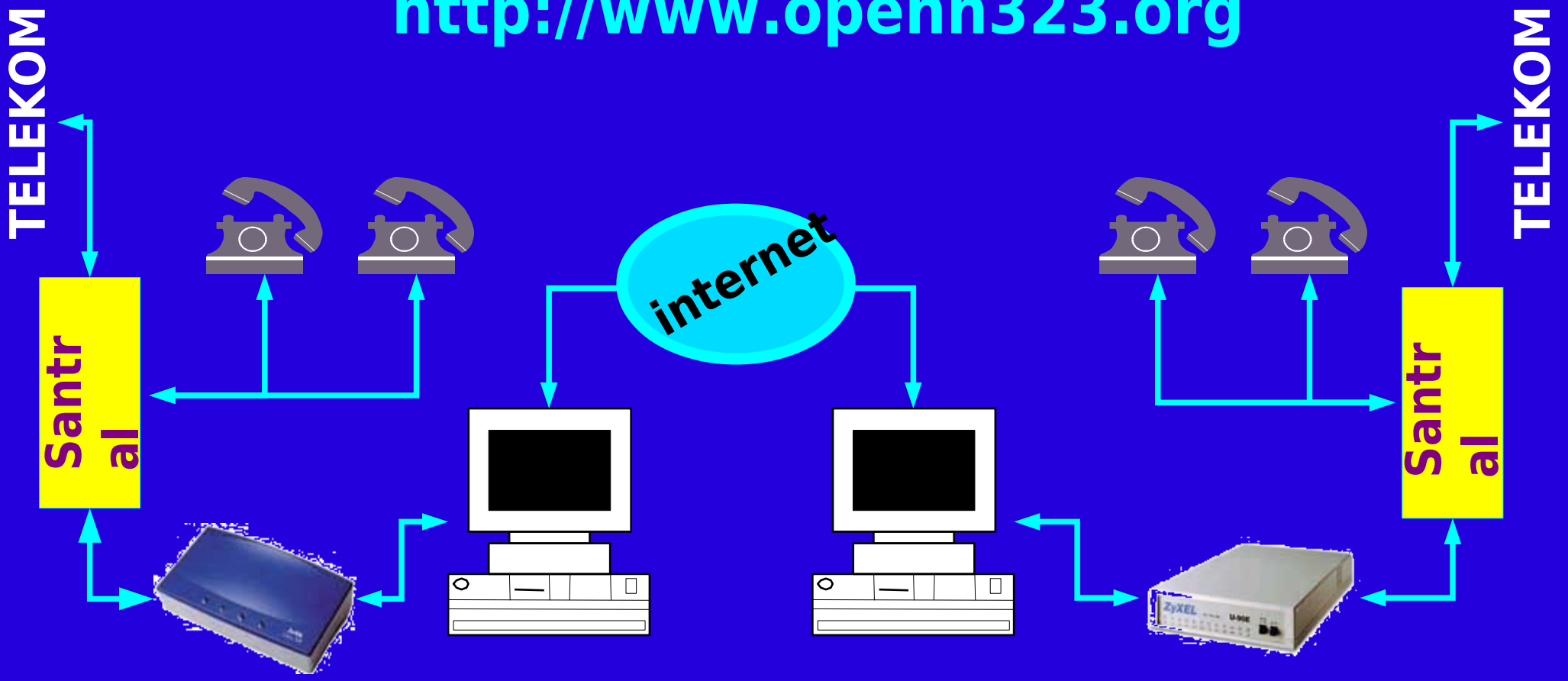
Yankı önleme gibi DSP yatkın işlevler.
Sıkıştırma işlevleri.
Compact Yapı.
H323 için hazır sürücü ve uygulamalar.

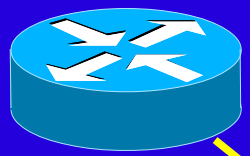
GnomePhone
GnomeMeeting
CU30

H323 VoIP uygulamaları

PSTNgw - PSTN Gateway
OpenH323GK - H323 Gatekeeper

http://www.openh323.org



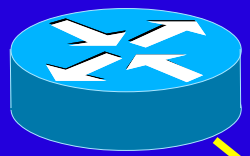


Bantgenişliği problemi

Tipik Telefon Bağlantısı 64KBit gerektirir
G723.1 sıkıştırma yöntemi 6.5 Kbps kullanır.
LCP-10 yöntemiyle bantgenişliği
ihtiyacı 2.5 Kbps düşürülebilir.

Çözüm için yapılabilecekler..

VoIP için gereken bantgenişliğini saklı tutun.
TOS bitleriyle paketlerinize öncelik isteyin.
Paketleri bir süre tutacak arabellek oluşturun.



Variable Latency problemi

- Paketler her zaman aynı sürede iletilmez.
- Paketin iletilmeye başlanması için beklenen süre (kuyruk zamanı) paketin iletim hızından çok farklı olabilir.
- Broadcast sistemlerinde (Uydular vs.) arabirim bazında ayrı bir kuyruk gecikmesi oluşur.

Çözüm için yapılabilecekler..

- MTU değerini düşürün.
- Paketleri bir süre tutacak arabellek oluşturun.