Bridging and (versus) routing

Given this setup…
... we want LEFT to ping RIGHT and get a reply

LEFT  MIDDLE  RIGHT

... 2 configs could make ‘em ping

1. Routing
   make 2 LANs out of it (2 broadcast domains)
   end-to-end connection achieved by routing the IP packets

2. Bridging
   make 1 consolidated LAN out of it (single broadcast domain)
   end-to-end by bridging the ethernet frames
Info’s usual trans-layer itinerary

Signals via hub ("layer 1 device")
Frames via bridge ("layer 2 device")

Packets via router ("layer 3 device")
Note, bridge scenario:
frame’s contained packet untouched

1. Two LANs –
routed connection end-to-end

Left Network – 200.2.2.150/24

Right Network – 192.168.3.0/24

© David Morgan 2007
Routed LANs – implementation

```
ifconfig eth0 192.168.3.1
route add -net 200.2.2.0/24 gw 192.168.3.1
```

```
ifconfig eth0 200.2.2.150
route add -net 192.168.3.0/24 gw 200.2.2.151
```

```
ifconfig eth1 200.2.2.151
route add -net 200.2.2.0/24 gw 192.168.3.1
```

Routed LANs – outcomes

```
ifconfig eth0 192.168.3.2
route add -net 200.2.2.0/24 gw 192.168.3.1
```

```
ifconfig eth0 200.2.2.150
```

```
route -n
```

```
Kernel IP routing table
```

```
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.3.0 0.0.0.0 255.255.255.0 U G 0 0 0 eth0
200.2.2.30 0.0.0.0 255.255.255.0 U 0 0 0 eth1
```

© David Morgan 2007
Routed LANs – outcomes

2. Single LAN – bridged connection end-to-end

Sole Network – 192.168.3.0/24

NOTE: physical NICs eth0/eth1 get no IP addresses
virtual NIC goldengate gets one (not two)
Bridged LAN – implementation

Bridged LAN – outcomes
Bridged LAN – outcomes

```
[root@emach4 ~]# ifconfig
et0      Link encap:Ethernet  HWaddr 00:60:08:96:AB:B2
    inet addr:192.168.3.99  Bcast:192.168.3.255  Mask:255.255.255.0
    Up  BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
    RX packets:185 errors:0 dropped:0 overruns:0 frame:0
    TX packets:81 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    Interrupt:18 Base address:0x6000
    Interrupt:5  Base address:0xd400
```

```
[root@emach4 ~]# route -n
Kernel IP routing table
Destination   Gateway         Genmask         Flags Metric Ref Use Iface
192.168.3.0  0.0.0.0         255.255.255.0   U   0   0  0  eth0
```

```
[root@emach4 ~]# route -n
Kernel IP routing table
Destination   Gateway         Genmask         Flags Metric Ref Use Iface
192.168.3.0  0.0.0.0  255.255.255.0   U   0   0  0  eth0
```

Bridged LAN – conceptual physical equivalent

```
1. eth0/eth1 interfaces pulled
2. goldengate interface inserted
3. hub fashioned from old eth0/eth1 hardware
```
Who’s sending to whom, and what - the fields that give a clue -

- ethernet sources and destinations
- IP sources and destinations
- IP’s header checksum
  “A checksum on the header only. Since some header fields change (e.g., time to live), this is recomputed and verified at each point that the internet header is processed.” rfc 791

- IP’s time-to-live
  “This field is modified in internet header processing. ...every module that processes a datagram must decrease the TTL by at least one...” rfc 791

- ICMP/ping’s message checksum

ing’s itinerary by router*

*data from Wireshark capture
ping’s itinerary by bridge

*data from Wireshark capture

A doubtful baseball analogy—
infield cutoff man

- Routing is like…
  - shortstop catches the ball from the outfielder
  - turns and makes a second, relay throw to the catcher
  - the throw caught by the catcher is the shortstop’s

- Bridging is like…
  - shortstop sees the outfielder’s throw strong and true
  - stands aside to let it go through on the bounce
  - the throw caught by the catcher is the outfielder’s
You don’t like my analogies? OK.

- Routing is like…
  - bridge machine catches the incoming packet
  - makes a new (slightly) modified one
  - puts it in a new frame that he makes and sends
  - the frame received by the destination endpoint is the bridge machine’s

- Bridging is like…
  - bridge machine is indifferent to the incoming frame
  - copies frame between interfaces without touching it
  - the frame received by the destination endpoint is the source endpoint’s

Don’t just take it from me…

“An ethernet bridge is a device commonly used to connect different networks of ethernets together, so that these ethernets will appear as one ethernet to the participants.

“Each of the ethernets being connected corresponds to one physical interface in the bridge. These individual ethernets are bundled into one bigger (‘logical’) ethernet, this bigger ethernet corresponds to the bridge network interface…

“Each bridge has a number of ports attached to it. Network traffic coming in on any of these ports will be forwarded to the other ports transparently, so that the bridge is invisible to the rest of the network…”

-- manpage for linux brctl command
Windows similarly - prebridge

2 interfaces (aka “adapters”) to start with:

Windows – creating the bridge

combine them into a bridge:
Windows - postbridge

new “Network Bridge” interface subsumes the 2 interfaces

Bridge “absorbs” interfaces