Proxy arp

Given this setup...

LEFT  MIDDLE  RIGHT

© David Morgan 2003-2008
... we want LEFT to ping RIGHT and get a reply

2 common techniques

- routing
- bridging
  separate discussion!

... and a more “non-standard” method
- proxy ARP
Proxied host – addressing

Significance of the addressing
( embedding )

Overall Network – 192.168.1.0/24
(ie, 255.255.255.0, size 256 from 0 thru 255)

Embedded subset – 192.168.1.240/28
(ie, 255.255.255.240, size 16 from 240 thru 255)
Proxied host – effect

MIDDLE proxying for RIGHT

- MIDDLE replies to LEFT (falsely) with his eth0’s hardware address
- LEFT proceeds to frame/send data (a ping) to MIDDLE/eth1
- LEFT arps for RIGHT, consequently
- RIGHT does not receive LEFT’s arp request but MIDDLE does
- MIDDLE replies to LEFT (falsely) with his eth0’s hardware address
- LEFT proceeds to frame/send data (a ping) to MIDDLE/eth1
- MIDDLE forwards to RIGHT (where tcpdump reveals)

```
$ ping 192.168.1.250
```

Proxied host – implementation

```
echo 1 > /proc/sys/net/ipv4/ip_forward
ifconfig eth1 192.168.1.241 netmask 255.255.255.240
ifconfig eth0 192.168.1.131 netmask 255.255.255.0
```

```
echo 1 > /proc/sys/net/ipv4/conf/eth0/proxy_arp
ifconfig eth0 192.168.1.198 netmask 255.255.255.0
ifconfig eth0 192.168.1.198/24
ifconfig eth0 192.168.1.250 netmask 255.255.255.240
```
Technically, we’re finished

- MIDDLE now arp-replies to LEFT on behalf of RIGHT
  - that’s proxy arp
  - mission accomplished
- Note – whenever they have anything to send the other
  - LEFT automatically arps for RIGHT, because RIGHT’s address falls within those LEFT considers “its own”
  - RIGHT does not automatically arp for LEFT, because LEFT’s address falls outside those RIGHT considers “its own”
- For left, proxy arp complements the existing arp LEFT will issue
- For right, there is no mechanism for reply
- If we want reply capability, we’re not finished

Achieving ping reply (technically, optional!)

- method A - routing the reply

```
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 1 > /proc/sys/net/ipv4/conf/eth0/proxy_arp
ifconfig eth1 192.168.1.241 netmask 255.255.255.240
ifconfig eth0 192.168.1.131 netmask 255.255.255.0
```

```
route add default gw 192.168.1.241
ifconfig eth0 192.168.1.198 netmask 255.255.255.0
```

*RIGHT won’t route to MIDDLE packets for remote LEFT unless told to do it. This tells it.
OR.. achieving ping reply *(technically, optional!)*

**method B – proxy-arping the reply**

```
echo 1 > /proc/sys/net/ipv4/ip_forward
        echo 1 > /proc/sys/net/ipv4/conf/eth0/proxy_arp
        ifconfig eth1 192.168.1.241 netmask 255.255.255.240
        ifconfig eth0 192.168.1.131 netmask 255.255.255.0
        echo 1 > /proc/sys/net/ipv4/conf/eth1/proxy_arp
```

```
echo 1 > /proc/sys/net/ipv4/conf/eth0/proxy_arp
route add host 192.168.1.198 eth0
```

```
ifconfig eth0 192.168.1.198 netmask 255.255.255.0
```

*RIGHT won’t arp for remote LEFT unless it thinks it’s local. This claims it is, prompts RIGHT to arp.

---

**Rules**

- Give IP addresses to both of proxying machine’s interfaces
- Create routes so proxying machine knows which hosts reside on left, and which on right
- Turn on proxy-arp on proxying machine’s interface(s) you wish to issue arp replies for other machines
- Arrange for proxied-for machine(s) to have a reverse path for any packets it wants to send to machines on far side of proxying machine

© David Morgan 2003-2008
Info

- see “man arp” under --s, and “man 7 arp”
- “Linux Advanced Routing and Traffic Control HOWTO,” Section 16.3 Pseudo-bridges with Proxy-ARP*
- “Guide to IP Layer Network Administration with Linux,” Section 9.3 Breaking a network in two with proxy ARP

* Instead of “ProxyARP Subnetting HOWTO,” which is outdated