Managing Multiple Internet Connections with Shorewall

Tom Eastep

Linuxfest Northwest

April 24-25, 2010

http://www.shorewall.net

Agenda

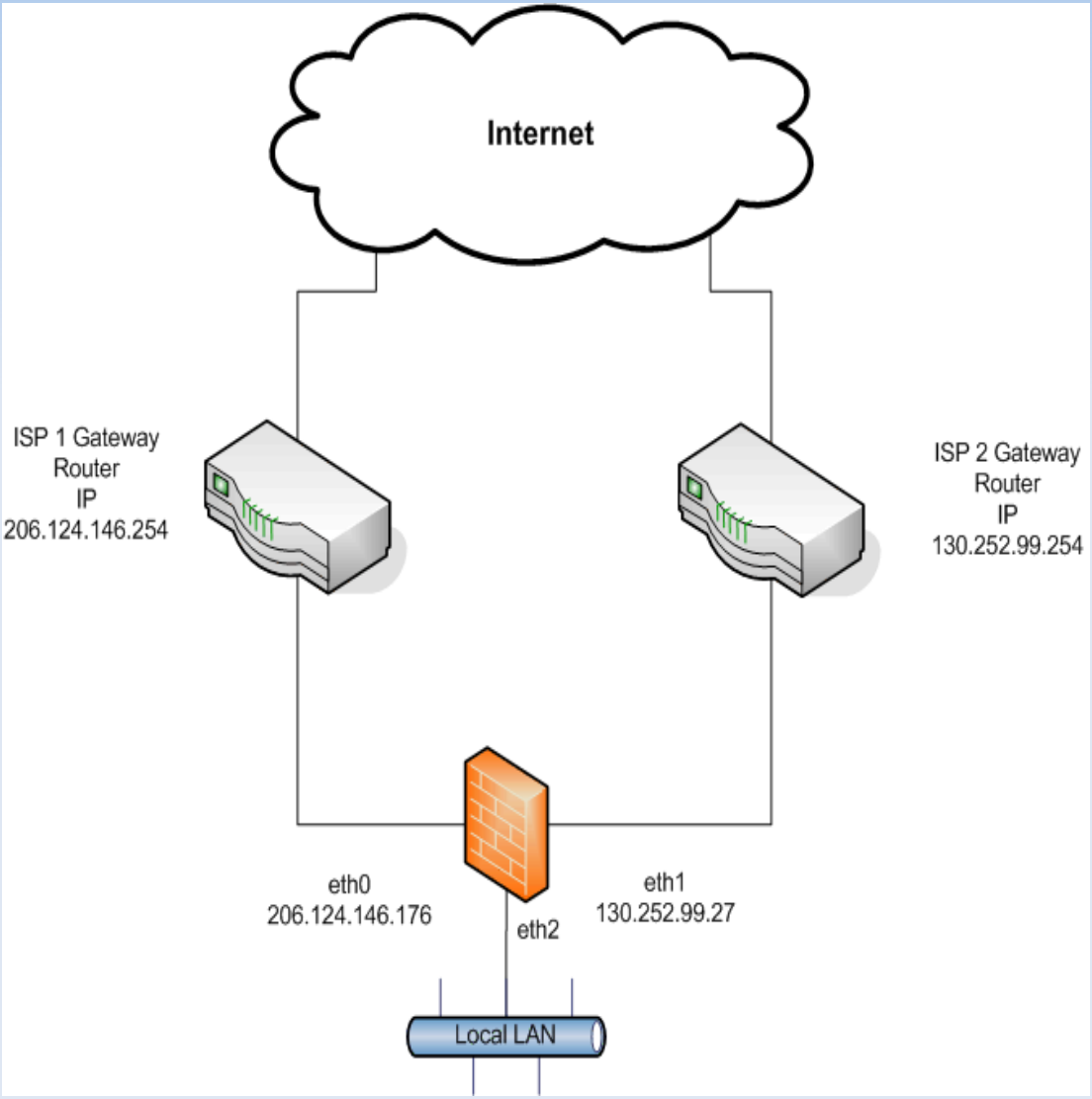
- Introduction
- Routing Refresher
- Introduction to Policy Routing
- Policy Routing and Shorewall
- Monitoring Link Status and Reacting to Failures
- Q&A

Introduction
About Me

- I am the creator of Shorewall
- I work for Hewlett-Packard
  - My job there has nothing to do with IP Networking
  - Shorewall development is not supported by HP
- This presentation is my own and does not represent HP in any way.

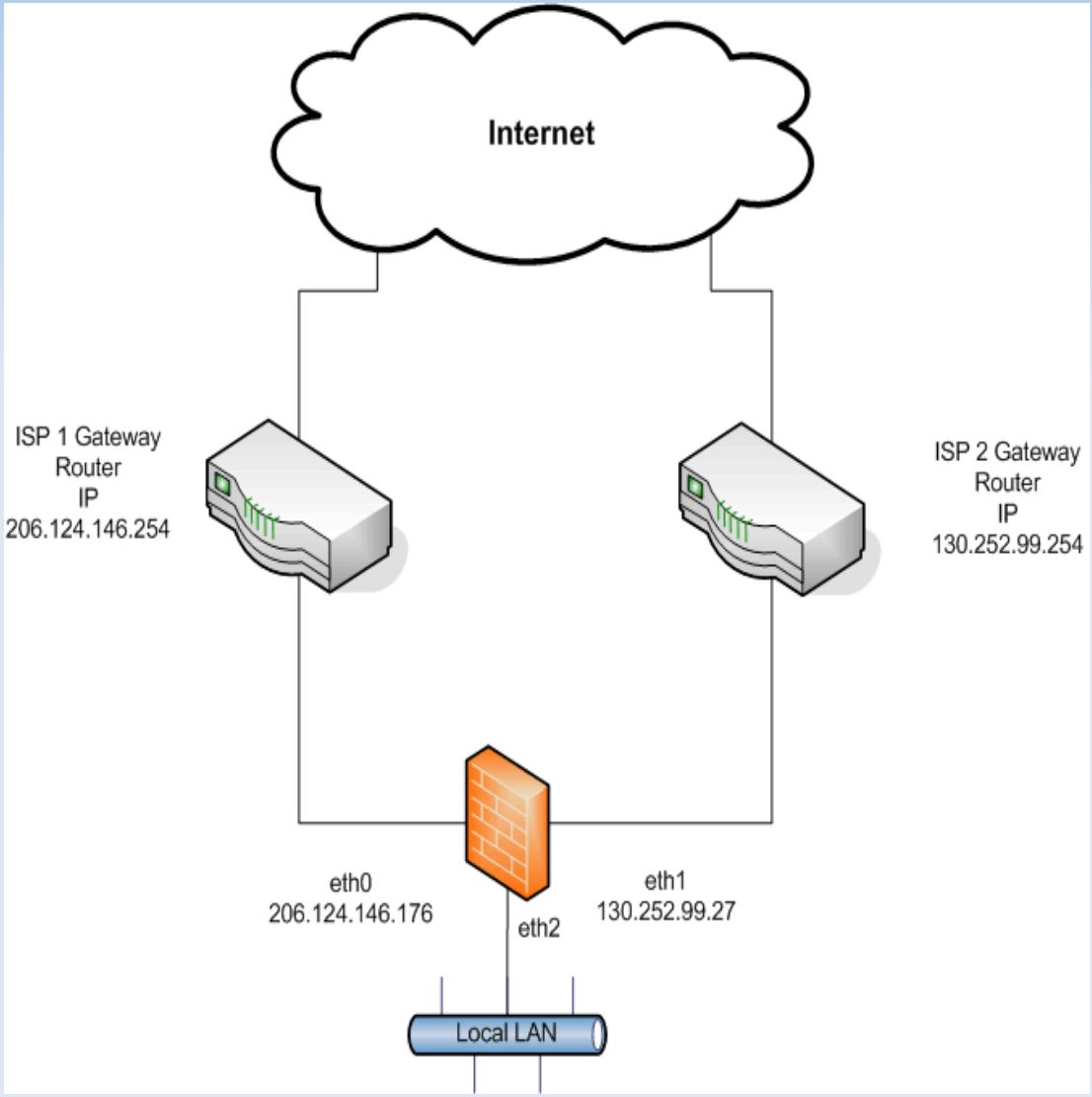
Introduction

- Having two or more internet links in a SOHO environment is becoming more common.
- Largely a routing problem



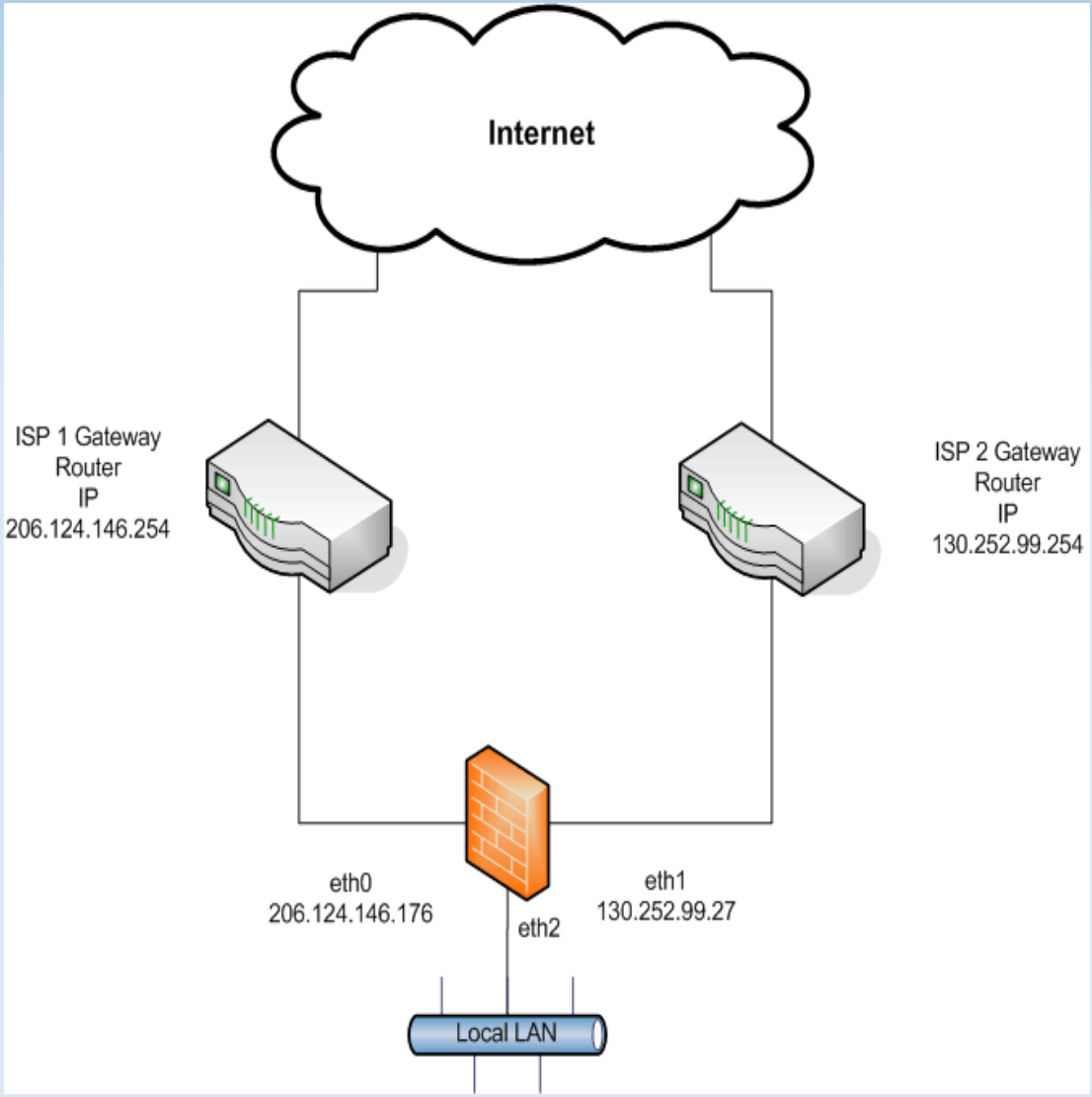
Introduction Requirements

- Optional load balancing
- Control which link to use for particular traffic
- Failover



Introduction
Requirements

- Shorewall can meet these requirements but a basic understanding of policy routing on Linux can make things go smoother.



Routing Refresher - Tables

- Routing is directed by a *routing table*
- A table entry contains (among other things):
  - Destination Network (may be a /32, in which case it is called a *host route*)
  - Interface
  - Gateway (Optional)
    - Usually includes a host IP address
    - If omitted, the destination network is connected directly to the interface.
    - Otherwise, specifies the next hop; packets are sent to the next hop using the Gateway's Layer 2 Address

Routing Refresher – Routing Tables

- Example from a Desktop Linux System

```
root@tipper:~# ip route ls
172.20.1.0/24 dev eth0 proto kernel scope global src 172.20.1.132 metric 1
169.254.0.0/16 dev eth0  proto kernel scope link metric 1000
default via 172.20.1.254 dev eth0 proto static
root@tipper:~#
```

- Ordered by most-specific to least-specific
  - ”via” denotes the gateway
  - ”default” is an alias for 0.0.0.0/0 which matches any IP address
  - First match determines routing

Routing Refresher – Routing Tables

- Point-to-point routes don't need to specify an IP address for the gateway:

```
default via dev ppp0
```

Routing Refresher
Routing Vs. Rules

- Routing determines where packets go
- Firewall filtering determines if they are allowed to go there or not
- DNAT firewall rules can change the destination address in packets but that occurs before routing (nat PREROUTING chain on Linux).
- SNAT firewall rules can change the routing of response packets.

Routing Refresher Response Packets

- Response packets are not like carrier pidgons – they don't instinctively know their way home
- Response packets can take a route other than the reverse of the corresponding request packet's route (asymmetric routing)
- The route taken by a response packet must be through all routers that perform DNAT on request packets.
- In the context of this talk, a response packet should go out the interface that the corresponding request arrived through

Policy Routing

- Linux Policy Routing
  - Multiple Routing Tables
    - Referred to by number and optionally by name
    - /etc/iproute2/rt_tables gives the correspondence between name and number
  - Routing Rules
    - Ordered by *priority*
    - If a packet matches a rule, it is routed using the related table
    - If not routed by that table, the next rule is tested
  - Always Enabled

Policy Routing - Example

- Example of Default /etc/iproute2/rt_tables

```
#
# reserved values
#
255     local
254     main
253     default
#
# local
#
```

- Example of Default Routing Rules

```
root@tipper:~# ip rule ls
0:      from all lookup local
32766:  from all lookup main
32767:  from all lookup default
root@tipper:~#
```

Policy Routing – Example Continued

- ”local” Routing Table

```
root@tipper:~# ip route ls table local
broadcast 127.255.255.255 dev lo  proto kernel  scope link  src 127.0.0.1
broadcast 172.20.1.0 dev eth0  proto kernel  scope link  src 172.20.1.132
local 172.20.1.132 dev eth0  proto kernel  scope host  src 172.20.1.132
broadcast 172.20.1.255 dev eth0  proto kernel scope link src 172.20.1.132
broadcast 127.0.0.0 dev lo  proto kernel  scope link  src 127.0.0.1
local 127.0.0.1 dev lo  proto kernel  scope host  src 127.0.0.1
local 127.0.0.0/8 dev lo  proto kernel  scope host  src 127.0.0.1
root@tipper:~#
```

- ”default” Routing Table is normally empty

```
root@tipper:~# ip route ls table default
root@typper:~#
```

Policy Routing – Routing Rules

- Routing rules allow assigning a packet to a table based on its firewall mark (fwmark) value (Netfilter's ”Swiss Army Knife”)
- Other routing keys are:
  - Input Interface
  - Source Address (net or host)
  - Destination Address (net or host)
  - TOS

Policy Routing – Balanced Routes

- Route with multiple next-hop gateways
  - Round-robin assignment by connection
  - Allows multiple default routes from a single system to be ”load-balanced”
  - Given that it is strictly round-robin, balancing isn't perfect.

Policy Routing – Balanced Routes

- Example 1 (ppp devices):

```
default
        nexthop via dev ppp0 weight 1
        nexthop via dev ppp1 weight 2
```

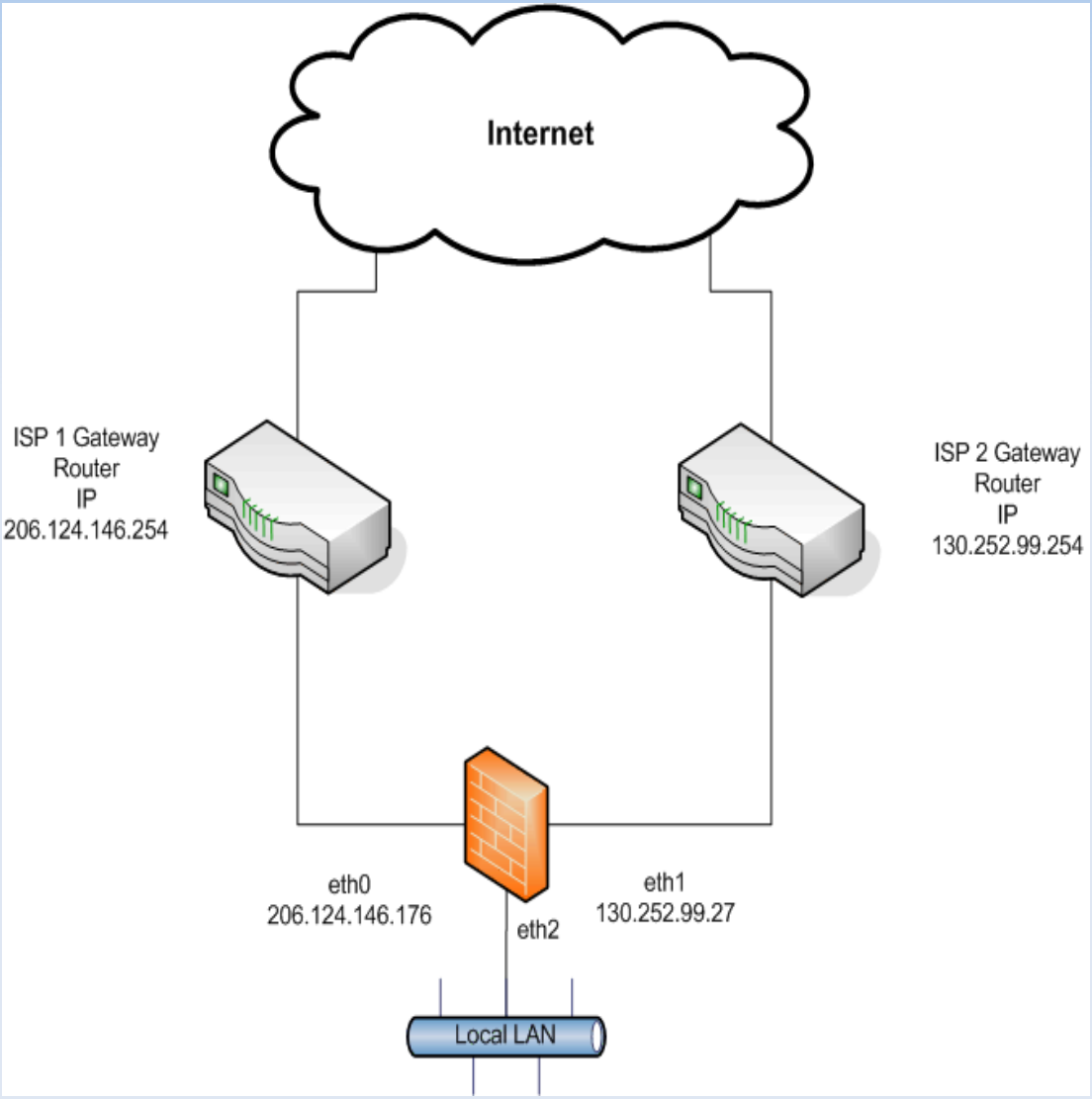
- Example 2

```
default
        nexthop via 10.253.0.254  dev eth1 weight 1
        nexthop via 66.146.173.97  dev eth2 weight 2
```

  - ”weight > 1” causes duplication of the route in the list so it is assigned more often.

Requirements

- Load balancing
- Failover
- Assign link to use for particular traffic



Shorewall and Policy Routing

- Shorewall *Multi-ISP* feature allows you to define:
  - Multiple Additional Routing Tables
  - Balanced Routes (with weights)
  - Assignment of particular connections to a specific table.
  - Geared toward multiple internet links from a single firewall/gateway
  - Other uses are possible

Shorewall Provider

- Each Shorewall (Internet Service) *Provider*:
  - Is normally associated with an internet link
    - Again, there are other uses described on the web site
  - Has it's own routing table
  - Defines a next-hop gateway
  - Typically has an fwmark value associated with it.
  - Defined by an entry in /etc/shorewall/providers
- Up to 255 providers may be defined

Shorewall Providers
Simple Example

- Two internet links *balanced*

```
#PROVIDER NUM MARK DUP   INTERFACE GATEWAY  OPTIONS        COPY
ISP1        1    1 main      eth0   detect  balance,track eth2
ISP2        2    2 main      eth1   detect  balance,track eth2
```

- MARK=1 causes any packet with firewall mark value 1 to be routed using the ISP1 routing table.
- 'DUP=main' causes the *main* routing table to be copied to create the provider's table
- 'COPY=eth2' means that only routes out of eth2 are to be copied.
- 'track' insures that a response goes out the interface that the corresponding request entered through

Shorewall Providers
Simple Example

```
#PROVIDER NUM MARK DUP   INTERFACE GATEWAY    OPTIONS         COPY
ISP1        1    1 main      eth0   detect    balance,track eth2
ISP2        2    2 main      eth1   detect    balance,track eth2
```

- 'balance' causes a balanced default route to replace the default route in the *main* table. 'balance=*n*' causes the nexthop weight to be set to *n*.
- 'track' locks connections to providers using connection marks. Can be made the default by setting TRACK_PROVIDERS=Yes in shorewall.conf.

Shorewall Providers
Simple Load-balancing Example

```
#PROVIDER NUM MARK DUP   INTERFACE GATEWAY  OPTIONS        COPY
ISP1        1    1 main      eth0   detect  balance,track eth2
ISP2        2    2 main      eth1   detect  balance,track eth2
```

- Generated Routing Rules

```
0:      from all lookup local
10001:  from all fwmark 1 lookup ISP1
10002:  from all fwmark 2 lookup ISP2
20000:  from addr-of-eth0 lookup ISP1
20256:  from addr-of-eth1 lookup ISP2
32766:  from all lookup main
32767:  from all lookup default
```

- ISP1 Routing Table

```
206.124.146.254 dev eth2  scope link  src 206.124.146.176
172.20.1.0/24 dev eth2 proto kernel scope link src 172.20.1.254
default via 206.124.146.254 dev eth2  src 206.124.146.176
```

Shorewall Providers - Selecting Provider for a Connection

- Although the routes are 'balanced', you still have control over which interface is used:
  - PREROUTING entries in /etc/shorewall/tcrules.
  - Use provider 1 except for outgoing mail and SSH

```
#MARK/          SOURCE      DEST          PROTO      DEST
#CLASSIFIER                                          PORT(S)
1:P             -           -
2:P             -           -             TCP        22,25
```

  - In this configuration, forwarded traffic never goes through the *main* routing table.
  - May also use /etc/shorewall/route_rules which will be mentioned again later

Shorewall Providers
VPN Issues

- The previous configuration works badly if the *main* table is being dynamically altered by VPN servers and clients because the provider tables don't get updated. Example: SSH to a host connected through a VPN.
- Solution 1: Add a route rule in /etc/shorewall/route_rules to route VPN traffic through the main table

```
#SOURCE                 DEST                    PROVIDER        PRIORITY
#
#OpenVPN clients
#
-                       172.20.0.0/24           main            1000
```

Shorewall Providers USE_DEFAULT_RT=Yes

- Solution 2: USE_DEFAULT_RT=Yes in shorewall.conf and use this provider configuration

```
#PROVIDER NUM MARK DUP   INTERFACE GATEWAY    OPTIONS         COPY
ISP1        1    1  -      eth0    1.2.3.4    track           -
ISP2        2    2  -      eth1    5.6.7.8    track           -
```

Shorewall Providers
USE_DEFAULT_RT

- Causes a balanced default route to be added to the *default* table rather than to the *main* table (balance=1 is the default for all providers)
- All traffic traverses the *main* table; even traffic marked to be routed out of a particular provider.
- Only works well with static gateways because the main table has no default route in it.
  - Dynamic IP management like DHCP wants to add a default route in the *main* table

Shorewall Providers
USE_DEFAULT_RT

- Routing Rules when USE_DEFAULT_RT=Yes

```
0:      from all lookup local
999:    from all lookup main
10000:  from all fwmark 1 lookup ISP1
10001:  from all fwmark 2 lookup ISP2
20000:  from addr-of-eth0 lookup ISP1
20256:  from addr-of-eth1 lookup ISP2
32767:  from all lookup default
```

Shorwall Providers
Fallback Providers

- shorewall.net configuration:
  - A fast cable link (Comcast) with a single dynamic IP address
  - A slower DSL link (Avvanta) with 5 static IP addresses
  - OpenVPN servers running on both
- Goal:
  - Use the Comcast link except where a static IP address is required or when the Comcast link is down.

Shorwall Providers
Fallback Providers

- Solution – make Avvanta a 'fallback' provider.

```
#PROVIDER NUM MARK DUP   IFACE GATEWAY    OPTIONS       COPY
Comcast     1    1 main eth0   detect     balance       eth2,\
                                                        venet0,\
                                                        tun+
Avvanta     2    2 main eth1   detect     fallback      eth2,\
                                                        venet0,\
                                                        tun+
```

- The dynamic default route for Comcast is in the *main* table (balance)
- The static default route for Avvanta is in the *default* table (fallback)
- TRACK_PROVIDERS=Yes in shorewall.conf causes 'track' to be assumed on both providers.

Shorwall Providers
Fallback Providers

- /etc/shorewall/route_rules:

```
#SOURCE                 DEST                    PROVIDER  PRIORITY
#
#OpenVPN clients
#
-                       172.20.0.0/24           main      1000
#
# Servers in OpenVZ containers - routes are generated by OpenVZ
#
-                       206.124.146.177         main      1001
-                       206.124.146.178         main      1001
#
# All 5 static IP addresses
#
206.124.146.176/30      -                       Avvanta   26000
206.124.146.180         -                       Avvanta   26000
```

Shorwall Providers
Fallback Providers

- Generated routing rules:

```
0:     from all lookup local                          <== Default
1001:  from all to 206.124.146.177 lookup main <== route_rules
1001:  from all to 206.124.146.178 lookup main <== route_rules
10000: from all fwmark 1     lookup Avvanta    <== track
10001: from all fwmark 2     lookup Comcast    <== track
20256: from 76.104.233.98 lookup Comcast       <== no 'loose'
26000: from 206.124.146.176/30 lookup Avvanta  <== route_rules
26000: from 206.124.146.180 lookup Avvanta     <== route_rules
32766: from all lookup main                    <== Default
32767: from all lookup default                 <== Default
```

- Note: 76.104.233.98 is the dynamic address of eth0

Dead Gateway Detection Failover

- Most SOHO ISP accounts don't offer routing protocol support.
- Linux lacks passive Dead Gateway Detection (DGD) without kernel patching.
- Solution – Active DGD (pinging)
  - Not a great choice but all we have 
  - Shorewall isn't a daemon so it can't do the pinging itself
  - Solution: Link Status Monitor (LSM)- http://lsm.foobar.fi/

Dead Gateway Detection Failover

- LSM keeps track of which interfaces are up and down and calls a user-provided script when an interface changes state.
- The *isusable* Shorewall extension script (user exit) can be used to help decide if the interface is up or down.
- The user-provider LSM script creates status files in /var/lib/shorewall/ which can be interrogated by the Shorewall *isusable* helper.

Dead Gateway Detection Failover

- The key is to arrange your Shorewall configuration such that if either of the interfaces fails or comes back up, a simple *shorewall restart -f* command will succeed
- On status change, LSM script re-writes status file(s) and does 'shorewall restart -f'
- Provider interfaces should be defined as *optional* in /etc/shorewall/interfaces
- When an optional provider is not available, its routing table and rules are deleted by restart.

Dead Gateway Detection
LSM Log

```
lsm: link Avvanta down event
lsm: name = Avvanta, replied = 80, waiting = 20, timeout = 20, late reply = 0, cons rcvd = 0, cons wait = 2, cons
miss = 2, avg_rtt = 126.305, seq = 7884
lsm: seq                                                                                                    *
lsm] used    11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
lsm: wait    00000000001110010001000000000000010000000000000000000000010101101110000000000100011000011001000001000
lsm: replied 11111111110001101110111111111111111011111111111111111111111110101001000111111111101110011111001101111101l
lsm: timeout 00000000001110010001000000000000010000000000000000000000010101101110000000000100011000011001000001000
lsm: error   00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

Dead Gateway Detection
LSM Email

```
Hi,

Connection Avvanta is now down.

Following parameters were passed:

newstate     = down
name         = Avvanta
checkip      = 206.124.146.254
device       = eth2
warn_email   = teastep@shorewall.net

Packet counters:
replied      = 80 packets replied
waiting      = 20 packets waiting for reply
timeout      = 20 packets that have timed out (= packet loss)
reply_late   = 0 packets that received a reply after timeout
cons_rcvd    = 0 consecutively received replies in sequence
cons_wait    = 1 consecutive packets waiting for reply
cons_miss    = 1 consecutive packets that have timed out
avg_rtt      = 150928 average rtt, notice that waiting and timed out packets have rtt = 0 when calculating this

Your LSM Daemon
```

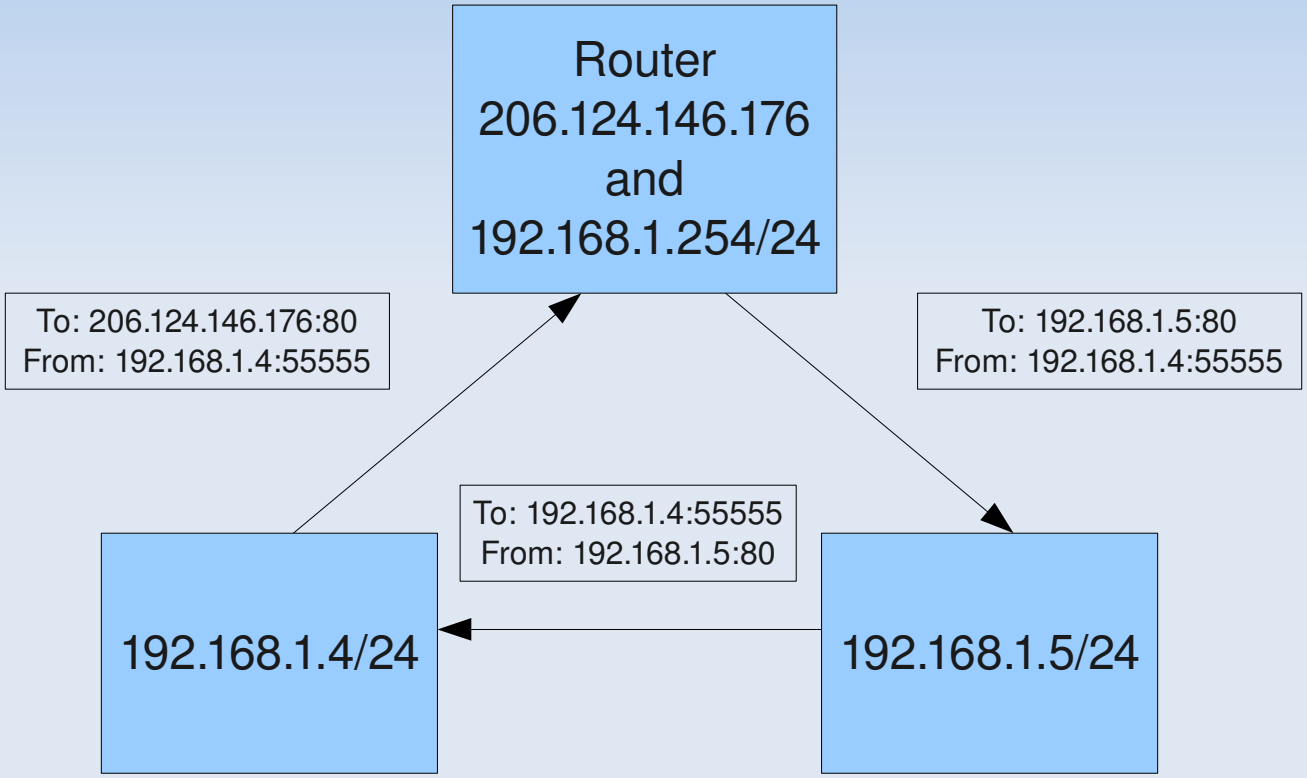
More Information

http://www.shorewall.net/MultiISP.html

Q & A

Backup Slides

Hairpinning Example (Why the NAT rule is Important)



Netfilter Packet Flow