Advanced routing scenarios

POLICY BASED ROUTING: CONCEPTS AND LINUX IMPLEMENTATION

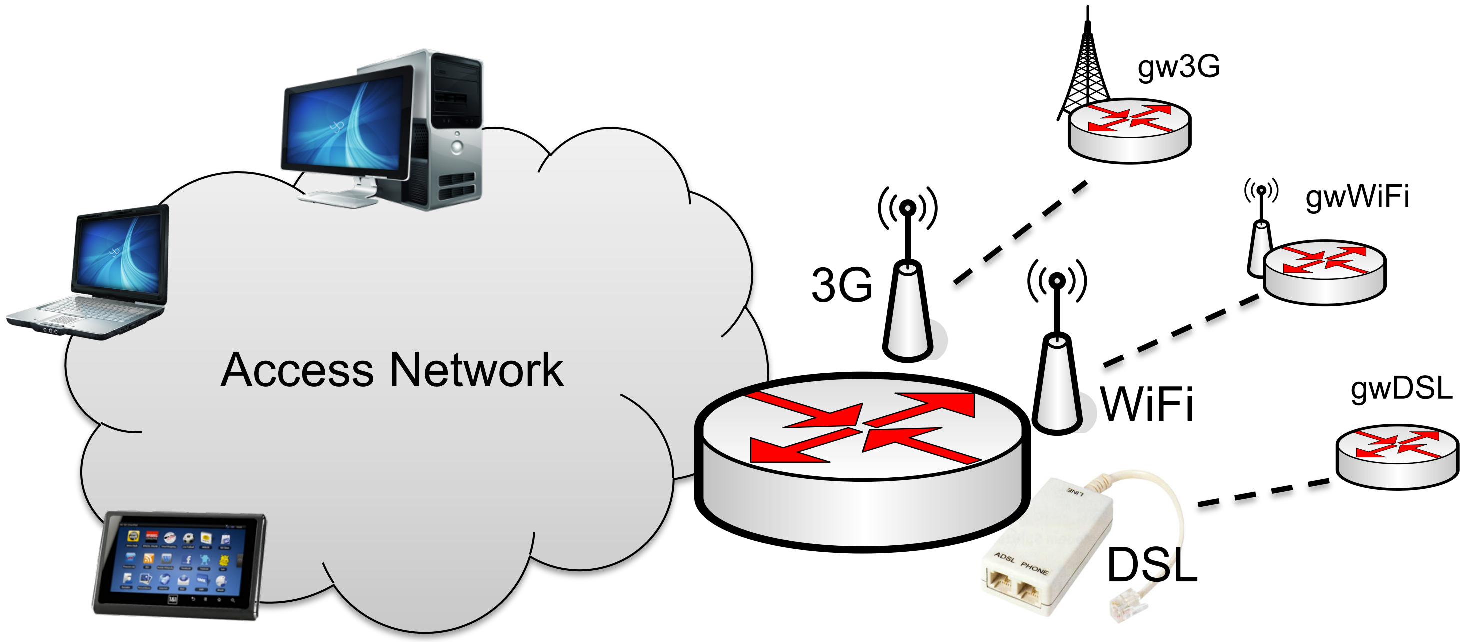
What is wrong with “standard” IP forwarding?

- The IP forwarding algorithm selects the route according to the destination in the IP packet header
- All traffic sent to the same destination (or destinations, in general: address/mask) will take the same route
- What if, for any reason, a portion of traffic addressed to the same destination(s) – but meeting some different criteria - needs to take a different path?
  - First “simple” example: real time traffic and background traffic have different service requirements and we might benefit from separating such types of traffic by forcing different paths. More later on…
- You may say, why don’t we aggregate traffic meeting the same criteria by using different multicast addresses?
  - FORGET MULTICAST if not in a local scenario!

What I can't do..

- What if I liked to take different routing strategies according to different purpose (besides the different destinations)?
- For example, I'd like to differentiate routing according to:
  - Source address (different (V)LAN attached to the same router)
  - Type of traffic (real time, delay tollerant, mission critical, etc…)
  - Size of the packet
  - Application protocols
  - And so on….

Example: multi-home/multi-provider access router



3 different access links (not very real scenario ☺)
DSL – hi reliability; flat, 1 Mbit/s
3G – medium reliability; expensive over 2 GB; 4 Mbit/s
WiFi – low reliability; free; 500 Kbit/s

How could I do the forwarding listed on the right with standard routing? Can I aggregate routes per destination?
NO! I should have the possibility to manage different rtables!

Type of traffic	Next hop	Iface
Real Time	gwDSL	DSL
WEB traffic up to 2GB	gw3G	3G
All from boss VLAN	gwDSL	DSL
Critical	gwDSL	DSL
P2P traffic	gwWiFi	WiFi
Remaining traffic	gwWiFi	WiFi

Solution: Policy Based Routing

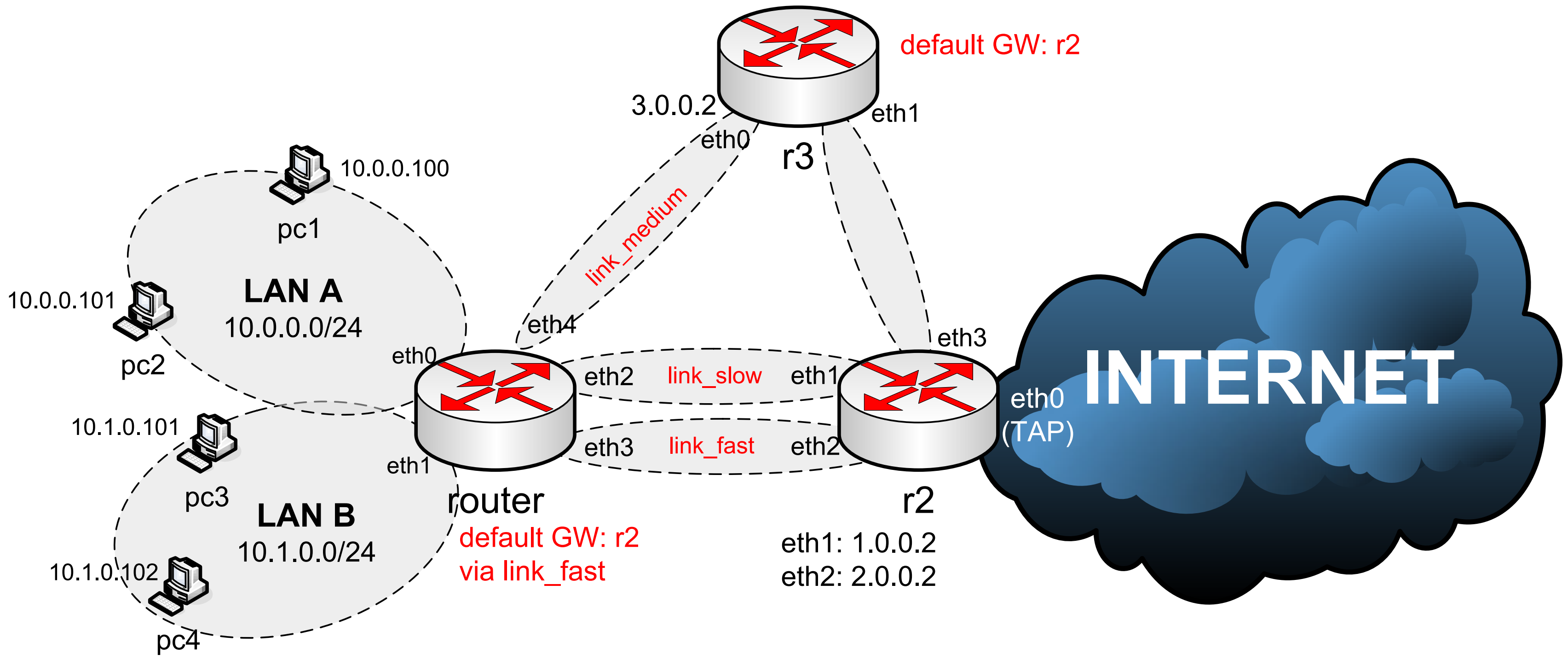
- Routers can be configured with different policies that selectively cause the packets to take different paths
- With PBR routers' IP/TCP stack manages multiple routing tables
- The specific table to be looked up is selected by a set of rules (or policies) configured by administrators
- The table selection is performed before the routing entry look-up
- Once the right table is selected, the forwarding algorithm goes on as in "standard" IP forwarding (i.e.: longest prefix match per-destination look-up)
- Policy description varies from OS to OS. In general we can take into account
  - Source addresses, ip.tos field, ip.protocol field, length, transport ports, more complex (implementation dependent) internal tagging system
  - Ex: traffic from the engineering department always take route1, real time traffic takes route2, traffic with len > 1500 takes route3, and so on…

PBR Benefits

- **Source-Based Transit Provider Selection:** Internet service providers and other organizations can use policy-based routing to route traffic originating from different sets of users through different Internet connections across the policy routers
- **Quality of Service (QOS):** Organizations can provide QOS to differentiated traffic by setting the precedence or type of service (TOS) values in the IP packet headers at the periphery of the network and leveraging queuing mechanisms to prioritize traffic in the core or backbone of the network
- **Cost Savings:** Organizations can achieve cost savings by distributing interactive and batch traffic among low-bandwidth, low-cost permanent paths and high-bandwidth, high-cost, switched paths
- **Load Sharing:** In addition to the dynamic load-sharing capabilities offered by destination-based routing that the Cisco IOS software has always supported, network managers can now implement policies to distribute traffic among multiple paths based on the traffic characteristics
- **Performance Improvement:** in case of links with different MTUs, packets can be selected according to the their length, so to avoid fragmentation

PBR in access routers

- PBR is useful for both access and core networks
- Due to the nature of this course, we'll use PBR in scenarios with routers with multiple access links
- Here's the reference lab: Lab7-pbr (just router's next hop addresses are shown)



Note about link bandwidth emulation

We approximately emulated the network bandwidth with NETFILTER and the limit module.

link_slow is limited by putting a limit rule (80 packets/second, burts 5) in FORWARD for packets with both input and output interface set to eth2.

Similarly, we set 800 packets/second with burst 150 for packets coming/going from/via eth4 for link_medium emulation.

We didn’t do anything for link_fast.

Anyway, this approach is really naïve and superficial. We could have used TC and the NETEM module to have better control over the queuing disciplines. We considered this approach out of the scope of this course. For any further info take a look at the two following links:
http://tldp.org/HOWTO/Traffic-Control-HOWTO/
http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

PBR and Linux

- Linux supports policy based routing. Set the following kernel configuration symbols
  - IP_ADVANCED_ROUTER=yes
  - IP_MULTIPLE_TABLES=yes
- With the symbols above the kernel is ready to
  - Manage (modify, create, delete) multiple routing tables (other then the default ones)
  - Specify policies pointing to one of the multiple tables
- Routing tables are defined in:
  `/etc/iproute2/rt_table`
- The policy routing system is managed with iproute2 (`ip rule`, `ip route`)

```
marlon@marlon-vmxbn:~$ cat /etc/iproute2/rt_tables
255 local
254 main
253 default
0   unspec

marlon@marlon-vmxbn:~$
```

The tables above are the default tables used by the kernel. When you don't specify the table for rule insertion, you are implicitly using the default table

Dumping the routing tables

`ip route show table “table_name”`

```
root@marlon-vmxbn:/# ip route show table main
default via 172.16.166.2 dev eth0  proto static
169.254.0.0/16 dev eth0  scope link  metric 1000
172.16.166.0/24 dev eth0  proto kernel  scope link  src 172.16.166.156  metric 1

root@marlon-vmxbn:/# ip route show table local
broadcast 127.0.0.0 dev lo  proto kernel  scope link  src 127.0.0.1
local 127.0.0.0/8 dev lo  proto kernel  scope host  src 127.0.0.1
local 127.0.0.1 dev lo  proto kernel  scope host  src 127.0.0.1
broadcast 127.255.255.255 dev lo  proto kernel  scope link  src 127.0.0.1
broadcast 172.16.166.0 dev eth0  proto kernel  scope link  src 172.16.166.156
local 172.16.166.156 dev eth0  proto kernel  scope host  src 172.16.166.156
broadcast 172.16.166.255 dev eth0  proto kernel  scope link  src 172.16.166.156

root@marlon-vmxbn:/# ip route show table default
```

Exactly like when didn’t know anything about PBR…
we just added `table “table_name”`

Managing new tables

To add a table just enter a line in /etc/iproute2/rt_tables formatted as:

“Index TableName”

Be sure not to use an index already used. The order in not important

For example:

```
root@marlon-vmxbn:/# echo “200 test” >> /etc/iproute2/rt_tables
```

Now we have a new routing table called “test”. How do we add routing entries to this table?
Just use “`ip route add|del`” as usual, but append the directive “`table test`”.

For example ($ADDR has to be compatible with the NIC addresses):

```
root@marlon-vmxbn:/# ip route add default via $ADDR table test

root@marlon-vmxbn:/# ip route del default table test
```

How do I force traffic to select a given routing table?

I.E.: how do I set the forwarding policies?

It’s time to set up the policies pointing to the new table.

To do so, we use the ip tool “rule”.

Let’s say we want force all traffic from address 10.0.0.23 to follow the table “test”

```
root@marlon-vmxbn:/# ip rule add from 10.0.0.23 table test
```

From this point on, all packets with ip.src == 10.0.0.23 will be forwarded according to the routing table “test”

Let’s say, we want to do so also for packets with ip.tos 3:

```
root@marlon-vmxbn:/# ip rule add tos 3 table test
```

Is there a order for rule verification?

Yes

Every rule has a priority (parameter “pri” in the command “ip rule” – in the previous example it was added automatically). The rule database is scanned in order of increasing priority.

Let’s dump the policy database

```
0:  from all lookup local
32763: from all tos reliability lookup test
32764: from 10.0.0.23 lookup test
32766: from all lookup main
32767: from all lookup default
```

Let’s add another rule between the 2nd and the 3rd

```
root@marlon-vmxbn:/# ip rule add iif eth0 pri 32765 table
test
```

which it will be applied to packet received from interface “eth0”

Some details omitted before…

At startup time the kernel configures the default RPDB consisting of three rules:

1. Priority: 0, Selector: match anything, Action: lookup routing table local (ID 255). The local table is a special routing table containing high priority control routes for local and broadcast addresses. Rule 0 is special. It cannot be deleted or overridden.
2. Priority: 32766, Selector: match anything, Action: lookup routing table main (ID 254). The main table is the normal routing table containing all non-policy routes. This rule may be deleted and/or overridden with other ones by the administrator.
3. Priority: 32767, Selector: match anything, Action: lookup routing table default (ID 253). The default table is empty. It is reserved for some post-processing if no previous default rules selected the packet. This rule may also be deleted.

ip rule (simplified) usage

ip rule add - insert a new rule

ip rule delete - delete a rule

- **from** *PREFIX*: select the source prefix to match.
- **to** *PREFIX:* select the destination prefix to match.
- **iif** *NAME*: select the incoming device to match. If the interface is loopback, the rule only matches packets originating from this host. This means that you may create separate routing tables for forwarded and local packets and, hence, completely segregate them.
- **oif** *NAME:* select the outgoing device to match. The outgoing interface is only available for packets originating from local sockets that are bound to a device.
- **tos** *TOS:* select the TOS value to match.
- **fwmark** *MARK:* select the fwmark value to match. priority PREFERENCE the priority of this rule. Each rule should have an explicitly set unique priority value. The options preference and order are synonyms with priority.
- **table** *TABLEID:* the routing table identifier to lookup if the rule selector matches. It is also possible to use lookup instead of table.

ip rule flush - also dumps all the deleted rules

ip rule show - list rules

PBR simple set-up

Specification

Traffic from LAN A → link_slow

Traffic from LAN B → link_fast

Traffic from LAN A, host 10.0.0.101 → link_fast

Suggested workflow

1) Define how many tables we need
2) Add and configure any extra routing table
3) Set the policies pointing to any extra table (attention to the rule priorities!!!)

Solution in file: Lab7-pbr/router/root/pbr1.sh

Solution

1) We need just one extra table as the default route is already via “link_fast” in the main table. Let’s create 1 table called “link_slow”
2) We need just the default route in table “link_slow” via 1.0.0.2
3) We need two policies (rule a with higher priority – i.e.: lover pri parameter):
   a. Packets from 10.0.0.101 look up table “main”
   b. Packets from 10.0.0.0/24 look up table “link_slow”

```
echo "200 link_slow" >> /etc/iproute2/rt_tables

ip route add default via 1.0.0.2 table link_slow

ip rule add pri 30000 from 10.0.0.101 table main
ip rule add pri 30001 from 10.0.0.0/24 table link_slow
```

Advanced netfilter/PBR interaction

- It looks like Linux PBR implementation is very limited as you can just consider source address, tos and incoming/outgoing interfaces….
- ….WRONG! You can combine the netfilter MARK with PBR rule definitions
- In other words, you can MARK packets in any possible way NETFILTER allows you to do and then select the proper routing policy by specifying the “fwmark” parameter
- Example: forward DNS packet over the slow link
  - With NETFILTER we can mark DNS packets with 100 (mark in PREROUTING)
  - With “ip rule” we select packets marked with 100 to look up link_slow table

```
router# iptables -A PREROUTING -t mangle -p udp --dport 53
-j MARK --set-mark 100

router# ip rule add pri 29000 fwmark 100 table link_slow
```

PBR complex set-up

Specification (with this priority order. i.e.: the first matching rule wins!)

DNS traffic → link slow

SSH traffic → link fast

TCP traffic with dports 40000:40100 → link_medium

Traffic from host 10.0.0.101 → link_medium

WEB traffic → link medium

Traffic from LAN A → link_slow

Traffic from LAN B → link_medium

Solution

- There are several solutions. The following this approach (in Lab7-pbr/router/root/pbr2.sh) is the first that came into my mind. It can be improved...
  - 3 marks: 1→slow, 2→medium, 3→fast
  - 2 tables: link_slow, link_medium (link fast is covered by the main table)
  - Mark everything with NETFILTER (keep the same order as in the spec)
  - To avoid multiple matching rules, always check if the packet is already marked

Solution

```
#create the extra tables
echo "200 link_slow" >> /etc/iproute2/rt_tables
echo "201 link_medium" >> /etc/iproute2/rt_tables

#add the default routes to the new tables
ip route add default via 1.0.0.2 table link_slow
ip route add default via 3.0.0.2 table link_medium

#add the routing policies. This time I don’t care about the order, as
netfilter will take care of the required priorities
ip rule add pri 20000 fwmark 1 table link_slow
ip rule add pri 21000 fwmark 2 table link_medium
ip rule add pri 22000 fwmark 3 table main
```

Solution

```
#set ENV variables
SLOW="-j mark --set-mark 1"
MEDIUM="-j mark --set-mark 2"
FAST="-j mark --set-mark 3"

#set the netfilter rules
iptables -A PREROUTING -m mark --mark 0 -p udp --dport 53 $SLOW
iptables -A PREROUTING -m mark --mark 0 -p tcp --dport 22 $FAST
iptables -A PREROUTING -m mark --mark 0 -p tcp -m multiport --dports
40000:40100 $MEDIUM
iptables -A PREROUTING -m mark --mark 0 -s 10.0.0.101 $MEDIUM
iptables -A PREROUTING -m mark --mark 0 -p tcp -m multiport --dports 80,433
$MEDIUM

#the last two policies (from LAN A and LAN B) can be configured with ip
rule "from" selector, with priority lower than the "fwmark" rules
ip rule add pri 23000 from 10.0.0.0/24 table link_slow
ip rule add pri 23001 from 10.1.0.0/24 table link_medium
```

Homework

- Create an access router with 3 links
- Forward with a round robin approach every new “connection” (for the contrack module) over the available links
- Create the LAB and the required scripts