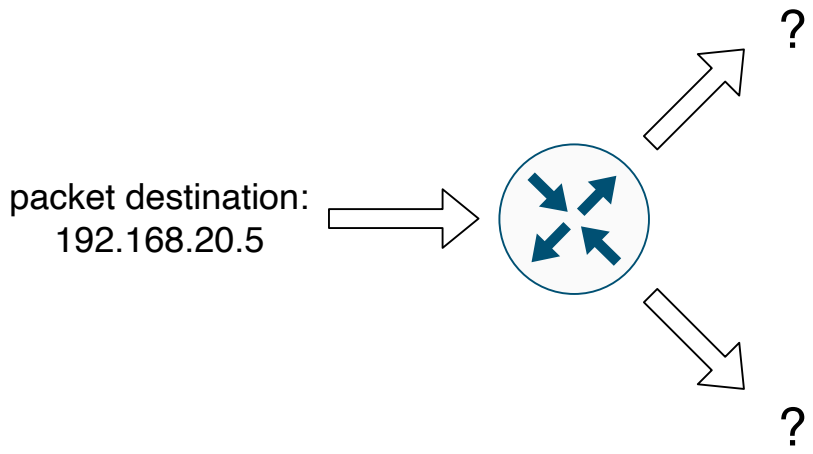
flokinet-009 - routing recap

Nicholas Morrison nick@nanocat.net

Routing



Static Routes

“For packets with destinations in this network, send them to this router.”

```
!
ip route   192.168.100.0/24      10.10.10.5
!               ^ destination        ^ router
```

- Receive a packet (dst: 192.168.100.8)
- Is there a matching entry in the routing table? (`show ip route`, or `show ip route 192.168.100.8`)
- Yes! `192.168.100.0/24 via 10.10.10.5`
- Forward that packet to the MAC address of `10.10.10.5`
- .. but how did the router decide “Yes!”?

Binary logic briefing

- Binary operators include AND, OR, NAND, NOR, XOR...
- Use them on binary values
- Examples with 1 bit:
  - `1 AND 1 == 1`
  - `1 AND 0 == 0`
  - `0 AND 0 == 0`
  - `1 OR  1 == 1`
  - `1 OR  0 == 1`
  - `0 OR  0 == 0`
- Two bits:
  - `11 AND 10 == 10`
  - `01 AND 00 == 00`
  - `11 OR  10 == 11`
  - `01 OR  00 == 01`

More binary logic examples

- Three bits:
  - `101 AND 110 == 100 (5 AND 6 == 4)`
  - `101 AND 101 == 101 (5 AND 5 == 5)`
  - `001 AND 101 == 001 (1 AND 5 == 1)`
  - `101 OR  110 == 111 (5 OR 6 == 7)`
  - `101 OR  101 == 101 (5 OR 5 == 5)`
  - `001 OR  101 == 101 (1 OR 5 == 5)`
- Four bits:
  - `1111 AND 1110 == 1110 (15 AND 14 == 14)`
  - `1101 AND 0001 == 0001 (13 AND 1 == 1)`
  - `1111 OR  1110 == 1111 (15 OR 14 == 15)`
  - `1101 OR  0001 == 1101 (13 OR 1 == 13)`
- .. but we really only care about **AND** at the moment.

ANDing for Quantisation with 8 bits

- Watch how ANDing these values with the value 252 (`11111100` in binary) quantises to multiples of 4:
  - `00000000 AND 11111100 == 00000000 (0 AND 252 == 0)`
  - `00000001 AND 11111100 == 00000000 (1 AND 252 == 0)`
  - `00000010 AND 11111100 == 00000000 (2 AND 252 == 0)`
  - `00000011 AND 11111100 == 00000000 (3 AND 252 == 0)`
- `0` through 3 when ANDed with 252 all came out as `0`.
  - `00000100 AND 11111100 == 00000100 (4 AND 252 == 4)`
  - `00000101 AND 11111100 == 00000100 (5 AND 252 == 4)`
  - `00000110 AND 11111100 == 00000100 (6 AND 252 == 4)`
  - `00000111 AND 11111100 == 00000100 (7 AND 252 == 4)`
- 4 through 7, when ANDed with 252, all came out as 4!

More ANDing for fun and quantisation

- Further..
  - `00001000 AND 11111100 == 00001000 (8 AND 252 == 8)`
  - `00001001 AND 11111100 == 00001000 (9 AND 252 == 8)`
  - `00001010 AND 11111100 == 00001000 (10 AND 252 == 8)`
  - `00001011 AND 11111100 == 00001000 (11 AND 252 == 8)`
- And these all resulted in 8 ^
- The numbers are being rounded down to the nearest multiple of 4.

  - (side note mega tip: 4 is also the value of the right-most “on” bit in the network mask...)
- This works all the way up to `11111111` (255)
  - `11111100 AND 11111100 == 11111100` (252 AND 252 == 252)
  - `11111101 AND 11111100 == 11111100` (253 AND 252 == 252)
  - `11111110 AND 11111100 == 11111100` (254 AND 252 == 252)
  - `11111111 AND 11111100 == 11111100` (255 AND 252 == 252)

ANDing and IPv4

- You can also do this with a whole IP address!
- It’s just 32 bits instead of 8
- the resulting quantised value is called the **Network Address**.
  - `11000000.10101000.00000000.00000101` (192.168.0.5)
  - `11111111.11111111.11111111.11111100` (255.255.255.252) (aka /30)
  - `AND`
  - `11000000.10101000.00000000.00000100` (192.168.0.4)
  - ==> the network address of 192.168.0.5/30 is 192.168.0.4
  - ==> 192.168.0.5 is inside the 192.168.0.4/30 network

HOWTO select a route

A packet arrives for forwarding with destination 192.168.100.1. Which route will be chosen?

```
r7#show ip route
...
Gateway of last resort is not set

 C        10.0.0.0/24 is directly connected, Ethernet1
 S        192.168.100.0/32 [1/0] via 10.0.0.5, Ethernet1
 S        192.168.100.0/30 [1/0] via 10.0.0.4, Ethernet1
 S        192.168.100.0/27 [1/0] via 10.0.0.3, Ethernet1
 S        192.168.100.0/24 [1/0] via 10.0.0.2, Ethernet1

r7#
```

- The **most specific** entry will take priority.
- This is also called the **longest match**.
- WAHT DOES IT MEAN

Most-specific match example: /32

- Our destination address, converted to binary:
- ```
  packet's dst:     192  .   168  .   100  .   1
                  11000000.10101000.01100100.00000001
  ```
  ```

- The first static route is 192.168.100.0 with a 32 bit netmask (255.255.255.255). Does it match?
- `S  192.168.100.0/32 [1/0] via 10.0.0.5, Ethernet1`

```
192.168.100.0  =>  11000000.10101000.01100100.00000000   <-+
                                                            |
192.168.100.1  =>  11000000.10101000.01100100.00000001     |
                      bitwise AND to find network           |
          /32  =>  11111111.11111111.11111111.11111111     |
                                equals                      |
                   11000000.10101000.01100100.00000001   <-+
```

- The result of ANDing the destination IP with the /32 netmask is **different** to the network of this static route.
- Therefore, the destination IP 192.168.100.1 is not inside the 192.168.100.0/32 network.

Most-specific match example: /30

- 30 bit netmask (255.255.255.252):
- `S  192.168.100.0/30 [1/0] via 10.0.0.4, Ethernet1`

```
192.168.100.0  =>  11000000.10101000.01100100.00000000   <-+
                                                            |
192.168.100.1  =>  11000000.10101000.01100100.00000001     |
                      bitwise AND to find network           |
          /30  =>  11111111.11111111.11111111.11111100     |
                                equals                      |
                   11000000.10101000.01100100.00000000   <-+
```

- The result of ANDing the destination IP with the /30 netmask is **the same** as the network of the static route.
- This route is a valid candidate!
- The **LENGTH** of this match is **30 bits**.

Most-specific match example: /27

- 27 bit netmask (255.255.255.224):
- `S  192.168.100.0/27 [1/0] via 10.0.0.3, Ethernet1`

```
192.168.100.0  =>  11000000.10101000.01100100.00000000   <-+
                                                            |
192.168.100.1  =>  11000000.10101000.01100100.00000001     |
                      bitwise AND to find network           |
          /27  =>  11111111.11111111.11111111.11100000     |
                                equals                      |
                   11000000.10101000.01100100.00000000   <-+
```

- The result of ANDing the destination IP with the /27 netmask is **the same** as the network of the static route.
- This route is a valid candidate!
- The **LENGTH** of this match is **27 bits**.

Most-specific match example: /24

24 bits of netmask (255.255.255.0):

- `S  192.168.100.0/24 [1/0] via 10.0.0.2, Ethernet1`

```
192.168.100.0  =>  11000000.10101000.01100100.00000000   <-+
                                                            |
192.168.100.1  =>  11000000.10101000.01100100.00000001     |
                      bitwise AND to find network           |
          /24  =>  11111111.11111111.11111111.00000000     |
                                equals                      |
                   11000000.10101000.01100100.00000000   <-+
```

- The result of ANDing the destination IP with the /24 netmask is **the same** as the network of the static route.
- This route is a valid candidate!
- The **LENGTH** of this match is **24 bits**.

The route is chosen.

- The three candidate matches for the destination 192.168.100.1 were:
  - 192.168.100.0/30 (30-bit match)
  - 192.168.100.0/27 (27-bit match)
  - 192.168.100.0/24 (24-bit match)
- longest match wins, so,
- `S  192.168.100.0/30 [1/0] via 10.0.0.4, Ethernet1` is the chosen route

HOWTO select a route: 192.168.100.25

A packet arrives for forwarding with destination 192.168.100.25. Which route will be chosen?

```
r7#show ip route
...
Gateway of last resort is not set

 C        10.0.0.0/24 is directly connected, Ethernet1
 S        192.168.100.0/32 [1/0] via 10.0.0.5, Ethernet1
 S        192.168.100.0/30 [1/0] via 10.0.0.4, Ethernet1
 S        192.168.100.0/27 [1/0] via 10.0.0.3, Ethernet1
```

```
 S        192.168.100.0/24 [1/0] via 10.0.0.2, Ethernet1

r7#
```

- The **most specific** entry will take priority.
- AKA the **longest match**.

Static route 192.168.100.0/32

- Our destination address, converted to binary:
- ```
  packet's dst:     192  .   168  .   100  .   25
                 11000000.10101000.01100100.00011001
  ```
- The first static route is 192.168.100.0 with a 32 bit netmask (255.255.255.255). Does it match?
- ```
   S        192.168.100.0/32 [1/0] via 10.0.0.5, Ethernet1

  192.168.100.0  =>  11000000.10101000.01100100.00000000   <-+
                                                              |
  192.168.100.25 =>  11000000.10101000.01100100.00011001     |
                        bitwise AND to find network           |
            /32  =>  11111111.11111111.11111111.11111111     |
                                  equals                      |
                     11000000.10101000.01100100.00011001   <-+
  ```
- The result of ANDing the destination IP with the /32 netmask is **different** to the network of this static route.
- Therefore, the destination IP 192.168.100.25 is not inside the 192.168.100.0/32 network.

Static route 192.168.100.0/30

- 30 bit netmask (255.255.255.252):
- ```
   S        192.168.100.0/30 [1/0] via 10.0.0.4, Ethernet1

  192.168.100.0  =>  11000000.10101000.01100100.00000000   <-+
                                                              |
  192.168.100.25 =>  11000000.10101000.01100100.00011001     |
                        bitwise AND to find network           |
            /30  =>  11111111.11111111.11111111.11111100     |
                                  equals                      |
                     11000000.10101000.01100100.00011000   <-+
  ```
- The result of ANDing the destination IP with the /30 netmask is **different** to the network of the static route.
- Therefore, the destination IP 192.168.100.25 is not inside the 192.168.100.0/30 network.
  ```

Static route 192.168.100.0/27

- 27 bit netmask (255.255.255.224):
- `S  192.168.100.0/27 [1/0] via 10.0.0.3, Ethernet1`

```
192.168.100.0  =>  11000000.10101000.01100100.00000000   <-+
                                                            |
192.168.100.25 =>  11000000.10101000.01100100.00011001     |
                      bitwise AND to find network           |
          /27  =>  11111111.11111111.11111111.11100000     |
                                equals                      |
                   11000000.10101000.01100100.00000000   <-+
```

- The result of ANDing the destination IP with the /27 netmask is **the same** as the network of the static route.
- This route is a valid candidate!
- The **LENGTH** of this match is **27 bits**.

Static route 192.168.100.0/24

24 bits of netmask (255.255.255.0):

- `S  192.168.100.0/24 [1/0] via 10.0.0.2, Ethernet1`

```
192.168.100.0  =>  11000000.10101000.01100100.00000000   <-+
                                                            |
192.168.100.25 =>  11000000.10101000.01100100.00011001     |
                      bitwise AND to find network           |
          /24  =>  11111111.11111111.11111111.00000000     |
                                equals                      |
                   11000000.10101000.01100100.00000000   <-+
```

- The result of ANDing the destination IP with the /24 netmask is **the same** as the network of the static route.
- This route is a valid candidate!
- The **LENGTH** of this match is **24 bits**.

The route is chosen.

- The two candidate matches for the destination 192.168.100.25 were:
  - 192.168.100.0/27 (27-bit match)
  - 192.168.100.0/24 (24-bit match)
- longest match wins, so,
- `S  192.168.100.0/27 [1/0] via 10.0.0.3, Ethernet1` is the chosen route

Summary

- the router uses the **network mask** to decide which network a given IP address is in
- ... by ANDing the IP address with the network mask
- the result is the **network address**.
- the router selects a route from the routing table to send packets
- it matches packets to routes by checking ...
- ... if the destination IP of the packet
- ... is within the network defined by the route.

Questions!