

# CSIS0234B Computer and Communication Networks (Class B)

## Reading for Tutorial 7

### Basic routing and subnetting in the Internet

In the last two tutorials, we examined how IP addresses are related with the hardware (Ethernet) addresses and the domain names. This time we will see how the IP addresses are used to facilitate routing, and how we can configure the computer to do routing the way we want.

#### 1. Routing in IP networks

In general, routing decisions might need to be made by the network layer of a computer because (a) the **transport** layer (TCP, UDP, etc) of the computer sends a segment, which is translated into packets that needs to be sent out; or (b) the **datalink** layer (network card) receives a frame, which is passed up to the network layer, although the destination IP address does not match the IP address, so the packet is to be **forwarded** instead.

Not all hosts will forward packets (in (b) above). For example, most computers with only one network interface will not forward packets at all. Even if the computer has two or more network interfaces, by default they have forwarding disabled<sup>1</sup>. A host with multiple interfaces but refuse to forward packets is called a **multi-homed host**. If you instead want a computer to act as a **router** (or “**IP gateway**”), you must explicitly enable forwarding. In Linux, the `root` user writes the string “1” to the file `/proc/sys/net/ipv4/ip_forward` to enable it:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Writing “0” instead of “1” will disable forwarding again.

#### 2. Routing tables concepts

Most hosts do very simple routing: if the destination host is on the local network, the packet is delivered to the destination host directly using the network interface. Otherwise, the data is sent to a **default gateway**, a “fallback” router to use whenever nothing more appropriate is found. It is more complicated if there are more than one routers which packets can be forwarded to. Different packets might need to be forwarded to different router as the **next hop**, according to the **network** that the destination host belongs to.

In each case, given the IP address of the destination host, we must be able to find the network it belongs to. As explained in the lecture, the IP address composes of some bits which is called the **network number**, and some bits which is called the **host number**. The network number determines whether a host is in a particular network. In the current **CIDR** scheme of IP addressing, given a destination IP address one cannot tell immediately how many bits of the address is the network number. Instead, the routing table entry must store both the network number and the netmask. Conceptually<sup>2</sup>, each incoming packet will be matched with all the routing table entries, in order to find the entries with a matching network number. There may be multiple entries that match a packet, and in such cases the entry with the most specific (i.e., largest) network mask prevails.

---

<sup>1</sup>This usually prevents a router with an incorrect routing table to create routing loops.

<sup>2</sup>In practice, the routing table rarely change, so the same address results in the same route. So a cache is kept to avoid having to scan the whole routing table to make the same decision everytime a packet arrives. You can examine the cache by using `route -C`.

### 3. Playing with the routing table

One can make things concrete by seeing and manipulating the routing table in a working computer. In Linux, we use the `route` command. We describe the simplest usage of the command, the full documentation can be found in its man page. Simply invoking `route` (usually with the `-n` option to avoid DNS lookup) shows the routing table<sup>1</sup>, e.g.:

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.0.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	192.168.0.1	0.0.0.0	UG	0	0	0	eth0

Each entry of the routing table has a `Destination` and a `Genmask` (i.e., netmask). Together they specify the network for which routing is specified. For example, the `192.168.0.0` and `255.255.255.0` above specify that the table entry is for the routing of destination addresses from `192.168.0.0` to `192.168.0.255`. If a packet has the destination `192.168.0.1`, both the first and the third entry match, but the first is chosen since it has a larger `Genmask`. It is also possible to have entries for a single host, by having `Genmask` being `255.255.255.255`.

When an address matches an entry in the table, the `Gateway` field tells how to reach the specified destination. If the `Gateway` field contains `0.0.0.0`, the destination is directly connected. Otherwise, it is the IP address of a router, which is used as the next hop.

Two of the remaining fields (`Use` and `Flags`) display some extra information about the route (e.g. In `Flag`, `U` indicates that the route is up, and `G` indicates that an external gateway is used, etc.). Other fields are not used.

The `route` command can also modify the routing table. It is invoked like this:

```
route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.0.1
```

The first argument is a keyword, either `add` or `del`, telling `route` either to add a new route or delete an existing one. The next argument `-net` specify that routing entry for a network, rather than a host, is installed. (For host routes it is not possible to use a netmask.) Then the destination, the netmask and the gateway (`gw`) are specified. Alternatively, one can specify the keyword `default`, which stands for `-net 0.0.0.0 netmask 0.0.0.0`. There are other configurable options, which you can find in the `route(8)` manpage.

### 4. Subnetting

Note that, in the above, the administrator of a computer will determine its own netmask. Nobody forces the administrator of all computers to use the same netmask, even for the same address—although if the netmask is not set correctly the routing might be done incorrectly. Indeed, different routers usually use different netmask for the same destination address, for the purpose of **aggregating** many networks into one. This saves time and memory when finding the routing tables automatically, and reduces the number of entries in the resulting routing tables.

For example, our university has the netmask `255.255.0.0` as seen from outside, for the network number `147.8.0.0`. Outside the university, routers would use this network number and network mask to perform routing. However, the routers within the university will use a finer mask. E.g., the CSIS department uses the mask `255.255.252.0`, with the network number

---

<sup>1</sup>The `netstat -r` command is used to examine the routing table on Solaris 8 systems.

147.8.176.0; while the EEE department uses the mask 255.255.255.0, with three network numbers from 147.8.180.0 to 147.8.182.0. So a subpart of the HKU IP addresses is given to the CSIS department and another subpart is given to the EEE department. Such subdivision of network addresses is called **subnetting**.

## 5. Finding the route to a particular host

Sometimes, especially when you are having trouble on your network setup, you want to know the path from your host to another host (and perhaps, where it fails). The IP header has an option field which, during the design of the Internet Protocol, is thought to handle this need: each router can add its record in an option of the IP header (the “record packet route” option). But it is soon found that the 40-bytes space for options in the IP header allows for at most 9 routers between a host and its destination, and thus is insufficient for today’s network size.

Instead, a completely different scheme is used by the standard utility `traceroute`, which performs such test. It tries to send packets (usually, UDP packets to random ports) to the target host, with a small TTL value  $n$  in the IP header. On the way to the destination, the TTL will drop to 0, and an ICMP (Internet Control Message Protocol) packet with type “Time-to-live exceeded” is sent back to the source host. If instead the TTL is large enough for the target host to be reached, an ICMP packet with type “Destination unreachable” (code “port unreachable”) is sent back to the host (hopefully the target is not using that UDP port). One can thus find the  $n$ -th router in the path from the host to the target. This is repeated from  $n = 1$  to successively larger  $n$  until the target is reached, and each such test is repeated a few times to measure the round-trip time (RTT) from the host to the  $n$ -th router and back. An example run of `traceroute` looks like this:

```
>traceroute hkueee.hku.hk
traceroute to hkueee.hku.hk (147.8.180.1), 30 hops max, 38 byte packets
 1 pcd-vta2-1-rx.netvigator.com (203.218.32.254)  22.066 ms  18.365 ms  20.157 ms
 2 w6002-v562.netvigator.com (218.102.57.198)   20.621 ms  19.391 ms  17.913 ms
 3 pcd507173.netvigator.com (218.102.39.173)    21.410 ms  18.357 ms  18.485 ms
 4 pccw-bsl2-GE.hkix.net (202.40.161.211)      22.184 ms  20.349 ms  18.881 ms
 5 202.40.217.2 (202.40.217.2)  27.396 ms  20.730 ms  21.959 ms
 6 147.8.239.1 (147.8.239.1)  30.313 ms  22.643 ms  29.078 ms
 7 147.8.240.228 (147.8.240.228)  22.137 ms  29.769 ms  21.577 ms
 8 147.8.240.234 (147.8.240.234)  30.244 ms  22.958 ms  20.762 ms
 9 147.8.240.218 (147.8.240.218)  31.315 ms  30.529 ms  28.914 ms
10 hkueee.eee.hku.hk (147.8.180.1)  31.859 ms * 30.459 ms
```

When reading the output of `traceroute`, keep in mind that routers usually have multiple addresses (one for each interface), and only one address (usually, the one closer to the host) is reported. E.g., in the above, step 9 (147.8.240.218) is probably already within the EEE department network 147.8.180.0/8, although only the IP address of the main university network is reported.

Of course this assumes that ICMP packets are not filtered by firewalls, which is usually the case (although not the case for our department).