15-441 Computer Networks **Lecture 15**
Mor Harchol-Balter **Oct 21, 2002**

Lecture 15: Subnetting, Supernetting, and IPv4

1 Announcements

1. Project 3 was assigned last Friday. Due 11/8/02.
2. Homework 3 will be assigned on Wednesday.

2 How to shrink the size of the forwarding tables

Last lecture we saw that routers within a local AS only have entries in their forwarding tables for those networks within the AS. In contrast, however, some routers (for example routers in the backbone) may have an entry for **every single network** in the Internet! The size of these forwarding tables can be a performance bottleneck. The purpose of this lecture is to discuss techniques which are used for keeping the forwarding tables from being as huge as we might think.

Within our discussion we will once again be discussing the format of a forwarding table. So far we've assumed that the forwarding table in every router in the Internet has entries of the form:

(Network Part of destination addr, Outgoing Interface, MAC addr of next hop)

We'll now see some circumstances where the entries in the forwarding table are a little more complicated than what's shown above.

Goal 1: It would be really nice if we could reduce the number of entries in the forwarding tables of regional and backbone routers. That is, we'd like to reduce the number of network addresses that these routers need to know of.

A seemingly separate problem has to do with IP addressing. It turns out that we are quickly **running out of IP addresses**. This may sound surprising, given that there are 4 billion possible IP addresses.

Question: Explain the problem of **inefficient** address utilization:

Answer:

Suppose you are a network with 2 nodes. This means that you need a class C network address. But now you've wasted 253 out of the 255 possible addresses associated with that network address!

Here's an even more serious example. Suppose you are a network with 256 nodes. You no longer fit into a class C network address, which means you need to ask for a class B network address. But you only use 256 out of the $2^{16} = 65,536$ IP addresses possible for that network address, a utilization of only 0.3%.

Question: How many class B network prefixes are there?

Answer: Only $2^{14} \approx 16000$ class B network prefixes. Thus it is easy to see how the class B addresses could get used up.

In fact it is exactly the class B addresses which are running out. (There are a lot more class C network addresses). This brings us to a second goal:

Goal 2: We'd like to use the address space more efficiently, so it doesn't run out.

We will discuss dual techniques: **subnetting** and **supernetting (a.k.a., CIDR)**, both of which aim to satisfy these two goals.

3 Subnetting

Subnetting aims to both reduce the total number of network numbers that are assigned, while also creating better utilization of the address space.

The idea is simple:

Under subnetting, a **single IP network number** (e.g., a class B network number) is allocated to several different networks. Each of these networks is now referred to as a **subnet**, and functions as a separate network (i.e., it is separated by routers from other networks).

For example, say have we have a class B network number: 128.96. This network number could correspond to a network with up to $2^{16} - 1$ nodes. Suppose that we don't have such a huge network. However we do have 3 smaller networks: Network X has $2^{15} - 1$ nodes. Network Y has $2^{14} - 1$ nodes. Network Z has $2^{14} - 1$ nodes. We would like to share the single class B network number among these 3 networks.

Here's how we might do this:

```
X =  1 0 0 0 0 0 0 0 . 0 1 1 0 0 0 0 0 . 1 x x x x x x x . x x x x x x x x

Y =  1 0 0 0 0 0 0 0 . 0 1 1 0 0 0 0 0 . 0 1 x x x x x x . x x x x x x x x

Z =  1 0 0 0 0 0 0 0 . 0 1 1 0 0 0 0 0 . 0 0 x x x x x x . x x x x x x x x
```

The 3 subnets are shown in Figure 1.

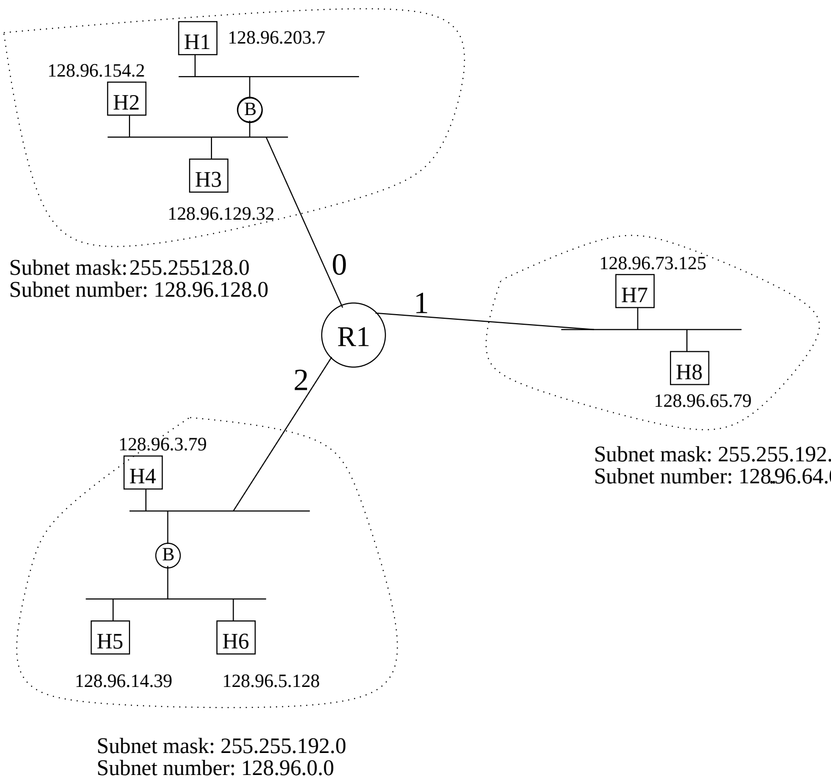

Figure 1: *3 subnets of network number 128.96.*

Question: But how will the routers know how to interpret IP addresses? Remember that the network number is still 128.96.

Question: For routers outside of network 128.96, what should the forwarding table entry look like?

Answer: These routers only need a single entry in their forwarding table:

(128.96, outgoing interface , MAC addr of next IP hop)

These networks see the collection of subnets as just the **single network**: 128.96.

Once the packet reaches a router with network number 128.96, that router needs to have a somewhat different forwarding table, which allows it to detect the appropriate subnetwork.

Question: How do you think the appropriate subnetwork is detected?

Answer: With a **mask**, or actually, with **multiple masks**.

Here's how it works. The router has a table which looks like this:

Mask	Subnet Number	Outgoing Interface
255.255.128.0	128.96.128.0	(Subnet X) Interface 0
255.255.192.0	128.96.64.0	(Subnet Y) Interface 1
255.255.192.0	128.96.0.0	(Subnet Z) Interface 2

The destination IP addr is ANDed with the first Mask entry. If the result matches the first Subnet Number, then the router knows that the destination is subnet X, and sends the packet there. If there is no match, then the destination IP addr is ANDed with the second Mask entry. If the result matches the second Subnet Number, then the router knows that the destination is subnet Y, and sends the packet there. If there is no match, then the destination IP addr is ANDed with the third Mask, and so on and so forth.

Observe that all the hosts on a given subnet are configured with the same mask. The bitwise AND of the host number and the mask defines the subnet number.

Question: Just to see if you really get it, suppose that we wanted to partition the addresses as follows: What would the router's table look like now?

```
X =  1 0 0 0 0 0 0 0 . 0 1 1 0 0 0 0 0 . 0 x x x x x x x . x x x x x x x x

Y =  1 0 0 0 0 0 0 0 . 0 1 1 0 0 0 0 0 . 1 0 x x x x x x . x x x x x x x x

Z =  1 0 0 0 0 0 0 0 . 0 1 1 0 0 0 0 0 . 1 1 x x x x x x . x x x x x x x x
```

Mask	Subnet Number	Outgoing Interface
255.255.128.0	128.96.0.0	(Subnet X) Interface 0
255.255.192.0	128.96.128.0	(Subnet Y) Interface 1
255.255.192.0	128.96.192.0	(Subnet Z) Interface 2

When a host wants to send a packet to a certain IP address, the first thing it does is to perform a bitwise AND between its own subnet mask and the destination IP address. If the result equals the subnet number of the sending host, then it knows that the destination host is on the same subnet and the packet can be delivered directly over the subnet. If the results are not equal, the packet needs to be sent to a router to be forwarded to another subnet.

Let's walk through an example:

Consider the subnets shown in Figure 2 below:

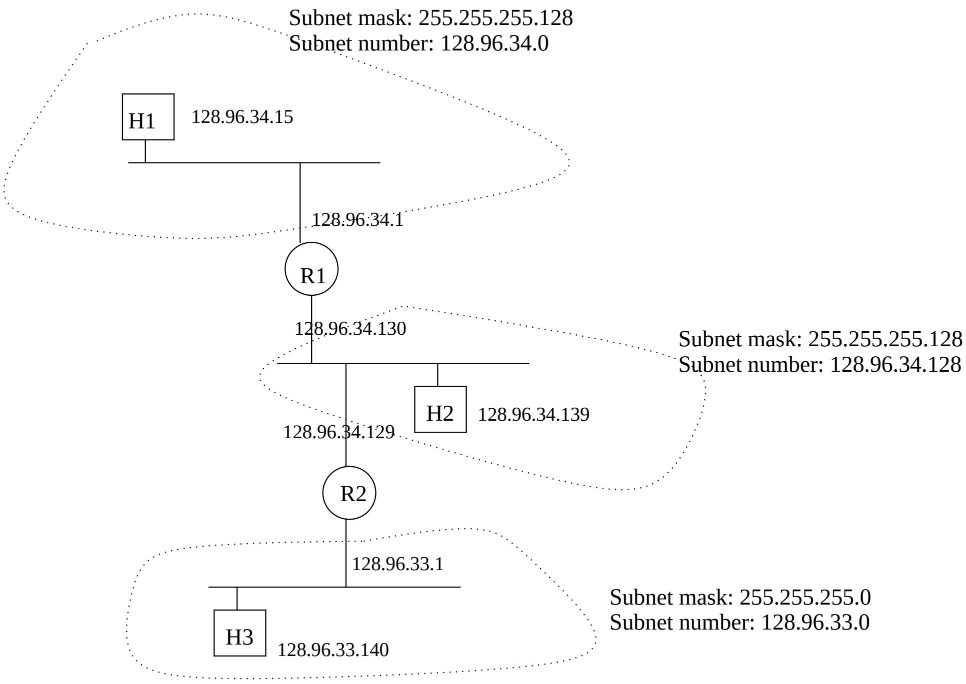

Figure 2: *Example involving forwarding between subnets.*

Suppose that H1 wants to send a datagram to H2. . H1 ANDs its subnet mask (255.255.255.128) with the IP address for H2 (128.96.34.139) to obtain: 128.96.34.128. This does *not* match the subnet number for H1 (128.96.34.0), so H1 knows that H2 is on a different subnet. Since H1 cannot deliver the packet to H2 directly over the subnet, it sends the packet to its default router R1. R1's forwarding table looks like this:

Mask	Subnet Number	Outgoing Interface	MAC addr of next IP hop
255.255.255.128	128.96.34.0	Interface 0	-
255.255.255.128	128.96.34.128	Interface 1	-
255.255.255.0	128.96.33.0	Interface 1	R2.mac

R1 then ANDs H2's address (128.96.34.139) with the subnet mask of the first entry (255.255.255.128) and compares the result (128.96.34.128) with the network number for that entry (128.96.34.0). Since this is not a match, it proceeds to the next entry. This time a match does occur, so R1 delivers the datagram to H2 using interface 1, which is the interface connected to the same network as H2.

An interesting comment before we leave the topic of subnetting: Note that in all the examples we've seen above, the 1's in the subnet mask were **contiguous**. This is *not* necessary, although it is advised.

Let's summarize some advantages, disadvantages, and general observations about subnetting:

1. Observe that under subnetting, we can end up with several routers all of which have the same network address (say 128.96) on their interfaces. Reaching the right network, 128.96, does not mean that we are now in datalink mode, because the right class B network may really correspond to many individual networks separated by routers.

2. Subnetting is typically done within a local AS. Here's what happens: The local AS goes to IANA to get some IP addresses to assign to its hosts. IANA can only give out class A or B or C addresses, so it chooses to give the local AS a class B address: 128.96.

   Question: Why does IANA prefer to give one IP address to a local AS, rather than a bunch of class C addresses?

   Answer: This way, the gateway router for the local AS only has to advertise one address: 128.96.

   However, the local AS may want to create several different networks. To do this, it uses subnetting to create many networks out of the single class B address. The forwarding tables of the routers within that local AS thus look a little different from ordinary forwarding tables because they use **masks**, which are set up by the **sys admin** of the local AS.

   Observe that for routers *outside* the local AS, *one entry* in their routing table suffices to reach the many subnetworks. This greatly reduces the size of their forwarding tables.

3. Subnetting cannot be done arbitrarily. The subnets need to be **physically close** to each other, because from far away they appear to be a single network.

4. Aside from making forwarding tables smaller (goal 1), subnetting also increases **utilization** of the IP address space (goal 2). However, there's room for further improvement, since portions of the class B network address 128.96 might still go unused by the local AS in the above example.

5. Subnetting has one other advantage: Suppose we had a very large network with network number 128.96, say. Suppose we didn't want everyone to be able to broadcast to everyone else. Subnetting would allow us to break out network into pieces (subnets) and put routers between the pieces, to provide **isolation** between the pieces. This is a useful administrative advantage. This was actually the original motivation behind doing subnetting.

   Question: Why not just use bridges to provide this isolation?

   Answer: Bridges can provide isolation, but they only understand MAC addresses, which are more cumbersome to manage since they aren't hierarchical.

4 Supernetting: CIDR

CIDR (pronounced "cider") – **Classless interdomain routing** – also aims to meet the two goals we describe above.

Let's return to the problem of **exhaustion of the class B addresses**. Recall that it is quite inefficient to assign a class B address to a network with only 256 hosts. One solution is to just stop giving out class B network addresses, and instead hand out the appropriate number of class C addresses.

Question: What's the problem with this?

Answer: Suppose a single AS has, say 16 class C network numbers assigned to it. This means that every BGP router needs to have 16 entries in its forwarding table for that AS, as compared with just a single class B entry.

CIDR attempts to **balance** the desire to minimize the number of entries in a router's forwarding table against the need to hand out addresses efficiently.

Question: Can you guess, based on the name, what CIDR does?

Answer: CIDR breaks the boundaries between address classes. It lets us imagine that we have for example a 20-bit network addresses (which sits between Class B and Class C).

Here's how it works:

Suppose that as in the example above we would like to hand out 16 class C addresses. However we want a single identifier which refers to all of them. Instead of handing out the class C address at random, we can hand out a block of **contiguous class C addresses**, e.g., 192.4.16 through 192.4.31.

Question: What does this block of addresses have in common?

Answer: The first 20 bits are the same for all these class C addresses:

```
1 1 0 0 0 0 0 0 . 0 0 0 0 0 1 0 0 . 0 0 0 1
```

Thus we have a **single network prefix**, a **20-bit number**, which can be used to refer to all of these 20 networks.

Question: How can a router deal with this 20-bit network number.

Answer: Unlike the case of classed addresses, it is not possible to determine the class of the network number by looking at the first couple bits. Thus the network number needs to be represented by a pair, denoting the network number and its length in bits:

$$(20\ ,\ \text{network number}\ )$$

This first entry is used in an identical fashion to a mask of 20 contiguous 1's.

Now consider what the forwarding table entries look like at a router. The table below shows an example:

Number bits	Network Number	Outgoing Interface	MAC addr of next IP hop
20	11000000.00000100.0001	Interface 0	R2.mac
23	11000000.00001000.0000111	Interface 1	R3.mac
17	11000000.00001000.1	Interface 2	R4.mac

Given a destination IP address, the router first does a bitwise AND between the destination IP address and 20 contiguous 1's. If the result matches the first network number, then the packet is sent out on Interface 0. Otherwise the router does a bitwise AND between the destination IP and 23 contiguous 1's, etc.. Observe that now we have totally changed the format of forwarding tables for some routers.

Eventually, the packet will make it to the appropriate "network," where the network might now be a particular 20-bit prefix. At this point, subnetting can be applied to the 20-bit prefix, to separate out the particular class C destination network.

Question: Recall that CIDR stands for Classless **Interdomain** routing. Why the word Interdomain?

Answer: The BGP (interdomain) routers might be doing **CIDR (a.k.a, supernetting)** and routers within an AS might do **subnetting**.

Question: One question that comes up in forwarding with classless addresses is what to do if our destination address matches more than one prefix, e.g. both 128.29 and 128.29.47 appear in the forwarding table of a single router. What do you recommend?

Answer: The rule is to use the **longest matching prefix**.

In summary:

In subnetting, we started with a standard network address, e.g. a class B network address, and attempted to **share/subdivide** this single address among multiple datalink networks. In supernetting, we start with a bunch of say class C networks, and **aggregate** them into a single address which is between a class B and C address. Both techniques have the same 2 goals.

5 Last resort: Forget IPv4, let's switch to IPv6

Despite active use of subnetting and CIDR (supernetting) in the Internet, we're still running out of IP addresses. Many people believe that we will eventually have to toss out IPv4 and switch to IPv6. We will discuss this next lecture. Before we do that, we'll need to discuss IPv4.

6 IPv4

We start with a description of the structure of **IP datagrams** under **IPv4** – IP version 4. Figure 3 shows the layout of each packet. (If you haven't noticed, packets are referred to as datagrams when we're in the network layer, just as they were referred to as segments when we were in the transport layer, and they will be referred to as frames when we get to the datalink layer.).

The first word of the header is pretty straightforward: The **version** header is used to specify version 4, as opposed to version 6, or whatever other versions might be invented later. The **header length** is specified in words (typically 5, assuming no options). The **TOS** field (type of service) is used to allow different quality of service (QOS) for different types of packets. For instance we might want to allow some packets to always move to the head of the line at every router queue. This is only starting to be deployed. We will talk a lot more about this later in the course when we teach QOS. The **Packet Length** field specifies the total length of the packet in bytes.

Question: What's the maximum size of an IP datagram?

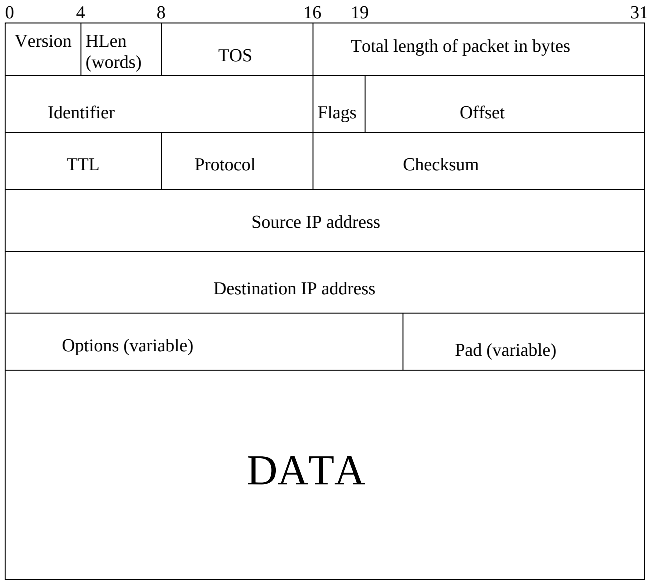

Figure 3: *IPv4 datagram*

Answer: 2^{16} bytes, although in practice datagrams are usually shorter than 1500 bytes.

We'll come back to the 2nd word of the header when we discuss fragmentation and reassembly.

In the third word of the header we see a **TTL** (time-to-live field), which is a hop-count, with typically a 64 default value.

Question: What do you think the **protocol** field is for?

Answer: This is the demultiplexing key that identifies the higher-level protocol to which this IP packet should be passed when it reaches its destination. This field may specify 'TCP' or 'UDP'.

The **checksum** field specifies the ones complement of the sum of the entire IP header viewed as a sequence of 2-byte numbers.

Question: If receiver sees that checksum is wrong, what does receiver do?

Answer: Drop the packet. In IP the motto is "Never hesitate to drop." Remember that the network layer is totally unreliable. For applications that need reliability, they use TCP above IP (hence the name TCP/IP).

Words 4 and 5 are the **IP addresses** of the source and destination.

Finally in word 6 there is an **options** field which is almost never used. This field was dropped from IPv6.

Fragmentation and Reassembly

Consider a packet hopping from router to router, as shown in Figure 4. The link-layer between router R1 and R2 may be able to carry frames up to 1500 bytes in size, whereas the link-layer between router R2 and R3 may be able to carry frames up to 500 bytes in size. Every datalink network has a **maximum transfer unit (MTU)** which is the largest *IP datagram* that it can carry in a frame (in its payload). The *TCP/IP Illustrated Vol I* lists MTU values for various datalink networks (see p. 30).

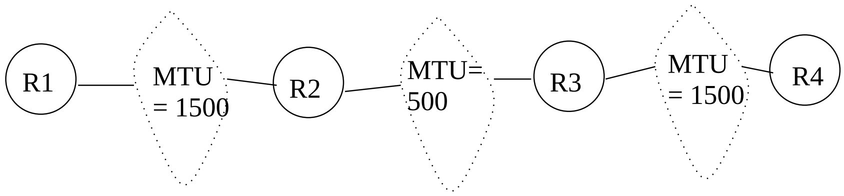

Figure 4: *Packet moves from router to router, but underlying link-layer MTU's might be different.*

Here's how this is dealt with:

The IP datagram picks a packet size that is equal to the MTU of its attached network, which it knows. However it doesn't know what networks it will be traversing on its way to the destination. When the packet gets to a router whose network MTU is too small to carry the packet, the packet is **fragmented** into smaller pieces.

Question: If we fragment a packet, what might we want to keep track of so as to allow the packet to be reassembled?

Answer: Each packet is assigned a **packet ID number**. When the packet is fragmented, all the fragments are assigned the same packet ID number. However, we need a way to **sequence** the fragments. To do this, we assign each fragment a **byte offset**. Finally, for convenience, there is a **flag** field which is used to specify whether this is the last fragment for the packet, or whether more are coming.

Figure 5 shows an example.

Question: Is reassembly done at each router, or only at the destination host?

Answer: Only at the destination host (simply a policy decision).

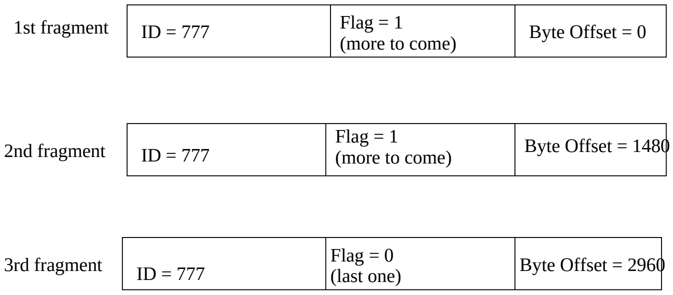

Figure 5: *Packet broken into 3 fragments. This figure shows the 2nd word of the header carried by each of the 3 fragments.*

Question: Why did we use the **byte offset** rather than just a sequence number in Figure 5?

Answer: A packet might be split several times on the way to its destination.

7 IPv6

Next time!

Thank you!