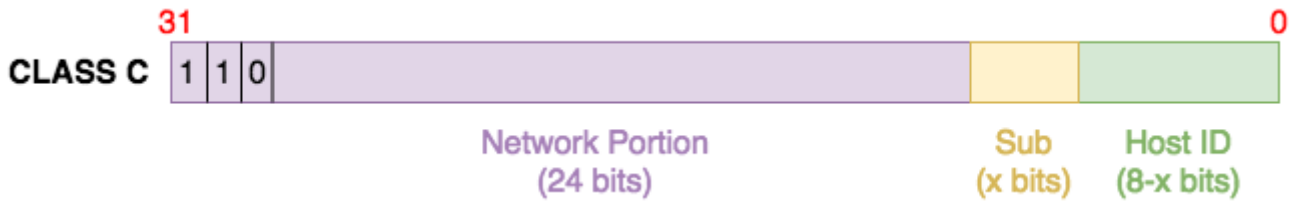
Network Basics

Subnetting Basics

Introduction to Subnets

Internet designers come with a solution to the issue of IP address wastage: Subnetting. Subnetting allows us to create a smaller network (sub-networks or subnets) inside a large network by borrowing bits from the Host ID portion of the address. We can use those borrowed bits to create additional networks, resulting in smaller-sized networks.



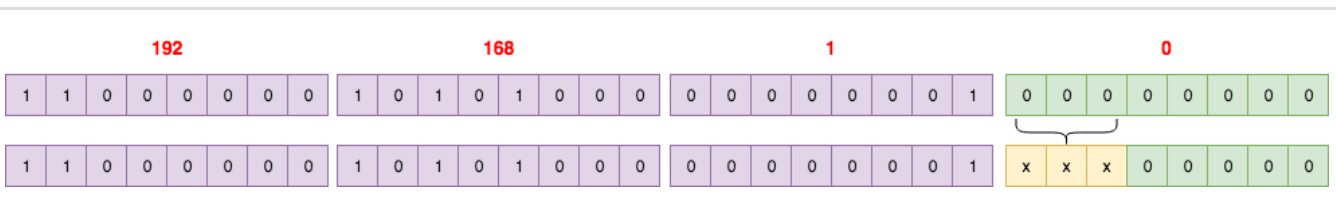
A Class C IP address

Imagine that we want to build four networks that will support up to 30 devices in different segments. Without subnetting, we will need four Class C IP addresses to support this design. For example:

```
Network #1: 192.168.1.0
Network #2: 192.168.2.0
Network #3: 192.168.3.0
Network #4: 192.168.4.0
```

With four Class C IP addresses we can subscribe 254 * 4 = 1016 hosts. But we have only 30 * 4 = 120 hosts. This means 1016 - 120 = 896 IP addresses will be wasted!

If you look at the design requirement of 30 hosts per network, you will see that 5 bits are enough to subscribe to 30 hosts in each network. And this means we still have 3 bits unused and with subnetting, we can use those three bits to create smaller networks. For this example, let's take the 192.168.1.0 network:



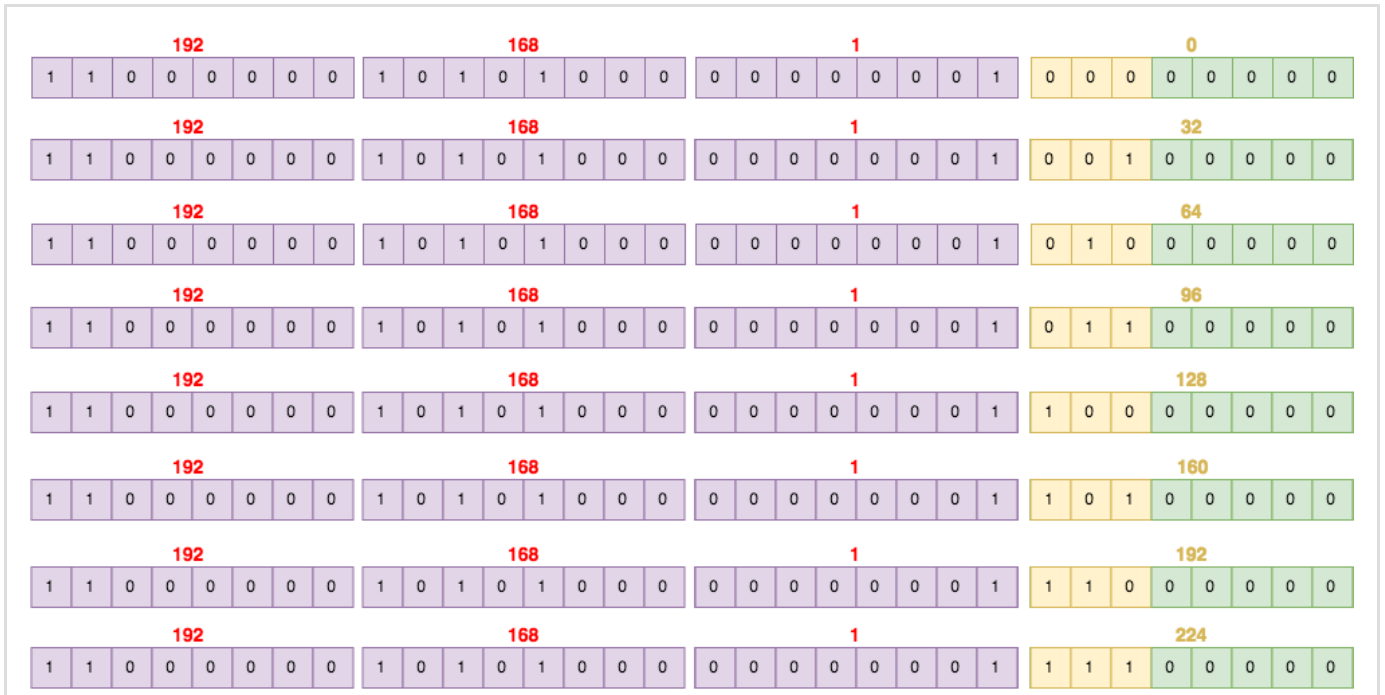
Borrowing bits from host segment of the IP address

By borrowing 3 bits, we can create 8 (2^3) subnets:

1. 192.168.1.0
2. 192.168.1.32
3. 192.168.1.64

4. 192.168.1.96
5. 192.168.1.128
6. 192.168.1.160
7. 192.168.1.192
8. 192.168.1.224

These subnet addresses look like normal IP addresses. However, looking at them in their binary form makes things clearer:



Subnet IP addresses

With subnetting, not only have we used only one Class C network, we have created 8 subnets from that network, each one supporting up to 30 hosts! We can use 4 of these subnets for our network and reserve the remaining 4 subnets for future expansion. This results in great waste reduction – from 896 wasted IP addresses to 120 reserved IP addresses.

Why do we need subnetting?

- **Reduce wastage:** As we have already seen, subnetting (and CIDR on a larger scale) helps us conserve IP addresses. While this is very important on the Internet (conserving public IP addresses), it is also useful on local area networks (LANs) where private IP addresses are used.

- **Improve Network Performance:** Subnetting improves the overall performance of a network. The larger a network is, the busier (more congested) it is. Consider the example of broadcasts – every host within an individual network will receive a broadcast even when it is not meant for them. This can affect performance especially during issues like broadcast storms. Therefore, the smaller the network, the more you can contain such issues within the subnet.

- **Isolation:** With smaller networks, you are able to isolate effectively as faults inside one subnet will not necessarily spread into other subnets. This is also important during security incidents so that even if one subnet is affected, the entire network is not brought down.

- **Easier administration:** Subnetting, when done properly, can make network administration more effective. For example, a multinational organization can design its network in such a way that each region is assigned an IP address block from a larger address block, and subnetting is used within regions to further divide the blocks among networks. This kind of design also improves routing as the routers in one region only need to know the “summarized” IP address block for other

regions rather than all the smaller IP address blocks. This reduces the size of the routing table and ensures that fluctuations in one region do not affect the entire network

Previous

Next

You have completed 0% of the lesson

0%

Jump to...

Network Basics

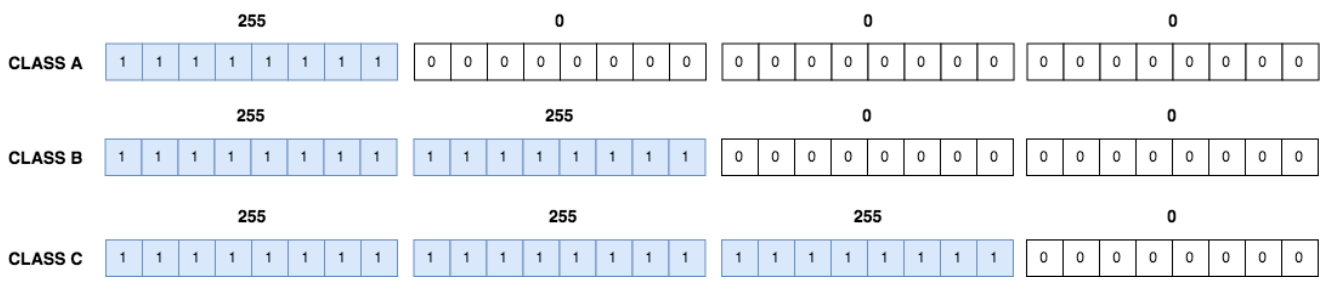
Subnetting Basics

Subnet Masks

With what we have done, we have created a problem for computers and other networking devices: how are they supposed to differentiate between a subnet `192.168.1.32` and an IP address `192.168.1.32`? This is where **subnet masks** (also called network masks) come in. A **subnet mask** is the representation of the network portion of an address. It is also made up of 32 bits with all the bits that represent the network portion being marked as **1s** and the other parts marked as **0s**.

For example, the default subnet masks of the IP address classes are:

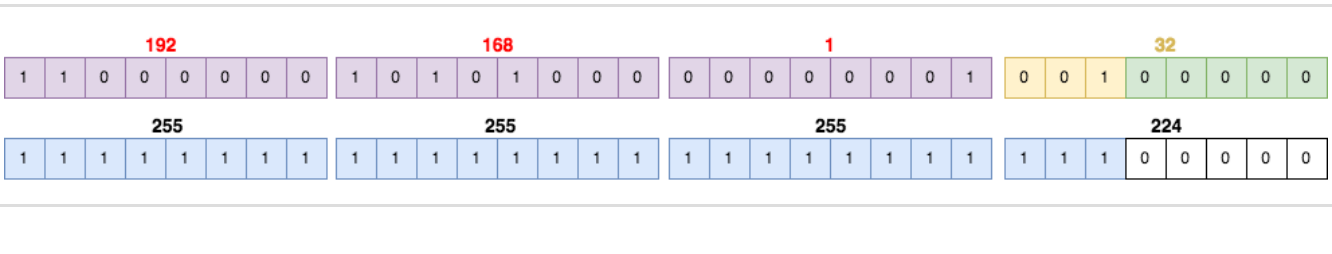
```
Class A: 255.0.0.0
Class B: 255.255.0.0
Class C: 255.255.255.0
```



Therefore, a Class C network of `192.168.1.0` can be represented as `192.168.1.0 255.255.255.0`.

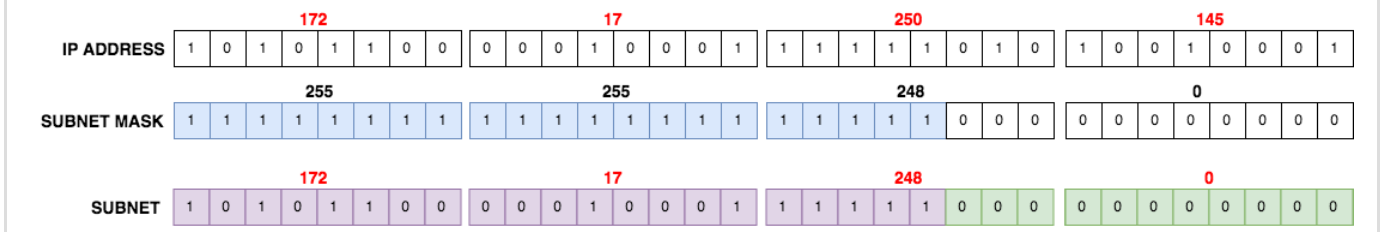
> **Tip:**
> - It can also be represented using prefix length (CIDR) notation where only the 1s that make up the network portion are counted and represented with a slash e.g. 192.168.1.0/24. We will talk about CIDR in the following lessons.

With subnetting, the borrowed bits from the host ID are counted as part of the network bits. So if we revisit our example above again, the `192.168.1.32` subnet can be represented as `192.168.1.32 255.255.255.224` (or `192.168.1.32/27`).



By comparing (logical AND) the “turned on” bits (i.e. 1s) in the subnet mask to an IP address, a network device can determine what network a particular IP address belongs to. For example, the `172.17.250.145` IP address with a

IP address belongs to. For example, the `172.17.250.145` IP address with a subnet mask of `255.255.248.0` belongs to the `172.17.248.0 255.255.248.0` subnet:



Previous

Next

You have completed 25% of the lesson

25%

Jump to...

Network Basics

Subnetting Basics

Classless Inter-Domain Routing (CIDR)

In order to reduce the wastage of IP addresses, a new concept of **Classless Inter-Domain Routing (CIDR)** is introduced.

When you receive a block of addresses from an ISP (Internet Service Provider), what you get will look something like this: `192.168.10.32/28`. This is telling you what your subnet mask is. The slash notation (/) means how many bits are turned on (1s). Obviously, the maximum could only be **/32** because a byte is 8 bits and there are 4 bytes in an IP address: 4 × 8 = 32. But keep in mind that the largest subnet mask available (regardless of the class of address) can only be a **/30** because you have to keep at least 2 bits for host bits.

Take, for example, a *Class A* default subnet mask, which is `255.0.0.0`. This means that the first byte of the subnet mask is all ones (1s), or `11111111`. When referring to a slash notation, you need to count all the 1s bits to figure out your mask. The `255.0.0.0` is considered a **/8** because it has 8 bits that are 1s—that is, 8 bits that are turned on.

A *Class B* default mask would be `255.255.0.0`, which is a **/16** because 16 bits are (1s):

```
11111111.11111111.00000000.00000000.
```

Rules for forming CIDR Blocks:

1. All IP addresses must be contiguous.
2. Block size must be the power of 2 (2^n). If the size of the block is the power of 2, then it will be easy to divide the Network. Finding out the Block Id is very easy if the block size is of the power of 2.
3. The first IP address of the Block must be evenly divisible by the size of the block. In simple words, the least significant part should always start with zeroes in Host Id. Since all the least significant bits of Host Id is zero, then we can use it as Block Id part.

Example:

Check whether `100.1.2.32` to `100.1.2.47` is a valid IP address block or not?

- All the IP addresses are contiguous.
- Total number of IP addresses in the Block = 16 = 2^4.
- 1st IP address: `100.1.2.00100000`

Since Host Id will contain the last 4 bits and all the least significant 4 bits are zero. Hence, the first IP address is evenly divisible by the size of the block. All three rules are followed by this Block. Hence, it is a valid IP

address block.

Previous

Next

You have completed 50% of the lesson

50%

Jump to...

Network Basics

Subnetting Basics

Variable Length Subnet Masks (VLSM)

So far, we have used subnetting to create fixed-size subnets e.g. four `/26` subnets from one `/24` block. However, the use of subnet masks and prefix lengths provides more flexibility – we can create subnets of varying sizes from the same address block i.e. VLSM.

Let us consider the following example. We are given a block of `172.16.1.0/24` and we need to split it such that the following requirements are met:

- A subnet that can accommodate 100 hosts
- A subnet that can accommodate up to 55 hosts
- Two subnets that can accommodate up to 12 hosts each

To solve this problem, start with the biggest block and keep going down. For example, we need a minimum subnet of `/25` to accommodate 100 hosts. (`/25` means we have 7 bits left for the host ID.) Therefore, we can split the `172.16.1.0/24` block into two subnets:

```
172.16.1.0/25
172.16.1.128/25
```

We can use the first subnet `172.16.1.0/25` for the 100 hosts leaving us with the other subnet, `172.16.1.128/25`.

The next largest subnet needs 55 hosts which can be accommodated with a `/26` subnet. This means we can split the `172.16.1.128/25` subnet into two smaller subnets:

```
172.16.1.128/26
172.16.1.192/26
```

We can use the `172.16.1.128/26` subnet for the network requiring 55 hosts leaving us with the `172.16.1.192/26` subnet to further break down.

The two other networks require 12 hosts meaning we need a minimum of `/28` subnets. Therefore, we can split the `172.16.1.192/26` subnet into 4 smaller subnets:

```
172.16.1.192/28
172.16.1.208/28
172.16.1.224/28
172.16.1.240/28
```

Therefore, our subnets are:

- `172.16.1.0/25` for the network with 100 hosts
- `172.16.1.128/26` for the network with 55 hosts
- `172.16.1.192/28` for the first network with 12 hosts
- `172.16.1.208/28` for the second network with 12 hosts

This means we still have two subnets (`172.16.1.224/28` and `172.16.1.240/28`) to use in the future.

Previous

Next

You have completed 63% of the lesson

63%

Jump to...

Network Basics

Subnetting Basics

Subnetting Example-1

255.255.128.0 (/17)

Let's take a look at our first example:

```
172.16.0.0 = Network address
255.255.128.0 = Subnet mask
```

- **Subnets?**

2^1 = 2 (same as Class C).

- **Hosts?**

2^{15} – 2 = 32,766 (7 bits in the third octet, and 8 in the fourth).

- **Valid subnets?**

256 – 128 = 128. 0, 128. Remember that subnetting in Class B starts in the third octet, so the subnet numbers are really 0.0 and 128.0, as shown in the next table. These are the exact numbers we used with Class C; we use them in the third octet and add a 0 in the fourth octet for the network address. To find the host addresses easily, initially write the subnet address and the broadcast address. After that, host addresses are between them and more clear to see.

- **Broadcast address for each subnet?**

The following table shows the two subnets available, the valid host range, and the broadcast address of each:

Subnet (do first)	0.0	128.0
First host (host addressing last)	0.1	128.1
Last host	127.254	255.254
Broadcast (do second)	127.255	255.255

Notice that we just added the fourth octet's lowest and highest values and came up with the answers. And again, it's done exactly the same way as for a *Class C* subnet. We just use the same numbers in the third octet and added 0 and 255 in the fourth octet.

Previous Next

You have completed 75% of the lesson

75%

Jump to...

Network Basics

Subnetting Basics

Subnetting Example-1

255.255.128.0 (/17)

Let's take a look at our first example:

```
172.16.0.0 = Network address
255.255.128.0 = Subnet mask
```

- **Subnets?**

2^1 = 2 (same as Class C).

- **Hosts?**

2^{15} – 2 = 32,766 (7 bits in the third octet, and 8 in the fourth).

- **Valid subnets?**

256 – 128 = 128. 0, 128. Remember that subnetting in Class B starts in the third octet, so the subnet numbers are really 0.0 and 128.0, as shown in the next table. These are the exact numbers we used with Class C; we use them in the third octet and add a 0 in the fourth octet for the network address. To find the host addresses easily, initially write the subnet address and the broadcast address. After that, host addresses are between them and more clear to see.

- **Broadcast address for each subnet?**

The following table shows the two subnets available, the valid host range, and the broadcast address of each:

Subnet (do first)	0.0	128.0
First host (host addressing last)	0.1	128.1
Last host	127.254	255.254
Broadcast (do second)	127.255	255.255

Notice that we just added the fourth octet's lowest and highest values and came up with the answers. And again, it's done exactly the same way as for a *Class C* subnet. We just use the same numbers in the third octet and added 0 and 255 in the fourth octet.

Previous Next

You have completed 75% of the lesson

75%

Jump to...

Network Basics

Subnetting Basics

Subnetting Example-2

255.255.255.224 (/27)

This time, we'll subnet the network address `192.168.10.0` and subnet mask `255.255.255.224`.

```
192.168.10.0 = Network address
255.255.255.224 = Subnet mask
```

- **How many subnets?**

224 is `11100000`, so our equation is $2^3 = 8$.

- **How many hosts?**

$2^5 - 2 = 30$.

- **What are the valid subnets?**

256 – 224 = 32. We just start at zero and count to the subnet mask value in blocks (increments) of 32: 0, 32, 64, 96, 128, 160, 192, and 224.

- **What's the broadcast address for each subnet (always the number right before the next subnet)?**

- **What are the valid hosts (the numbers between the subnet number and the broadcast address)?**

To answer the last two questions, first, just write out the subnets, and then write out the broadcast address—the number right before the next subnet. Last, fill in the host address. The following table gives you all the subnets for the `255.255.255.224` Class C subnet mask:

The subnet address	0	32	64	96	128	160	192	224
The first valid host	1	33	65	97	129	161	193	225
The last valid host	30	62	94	126	158	190	222	254
The broadcast address	31	63	95	127	159	191	223	255

Previous

End of lesson

Jump to...

Network Basics

Network Address Translation (NAT)

Network Address Translation (NAT)

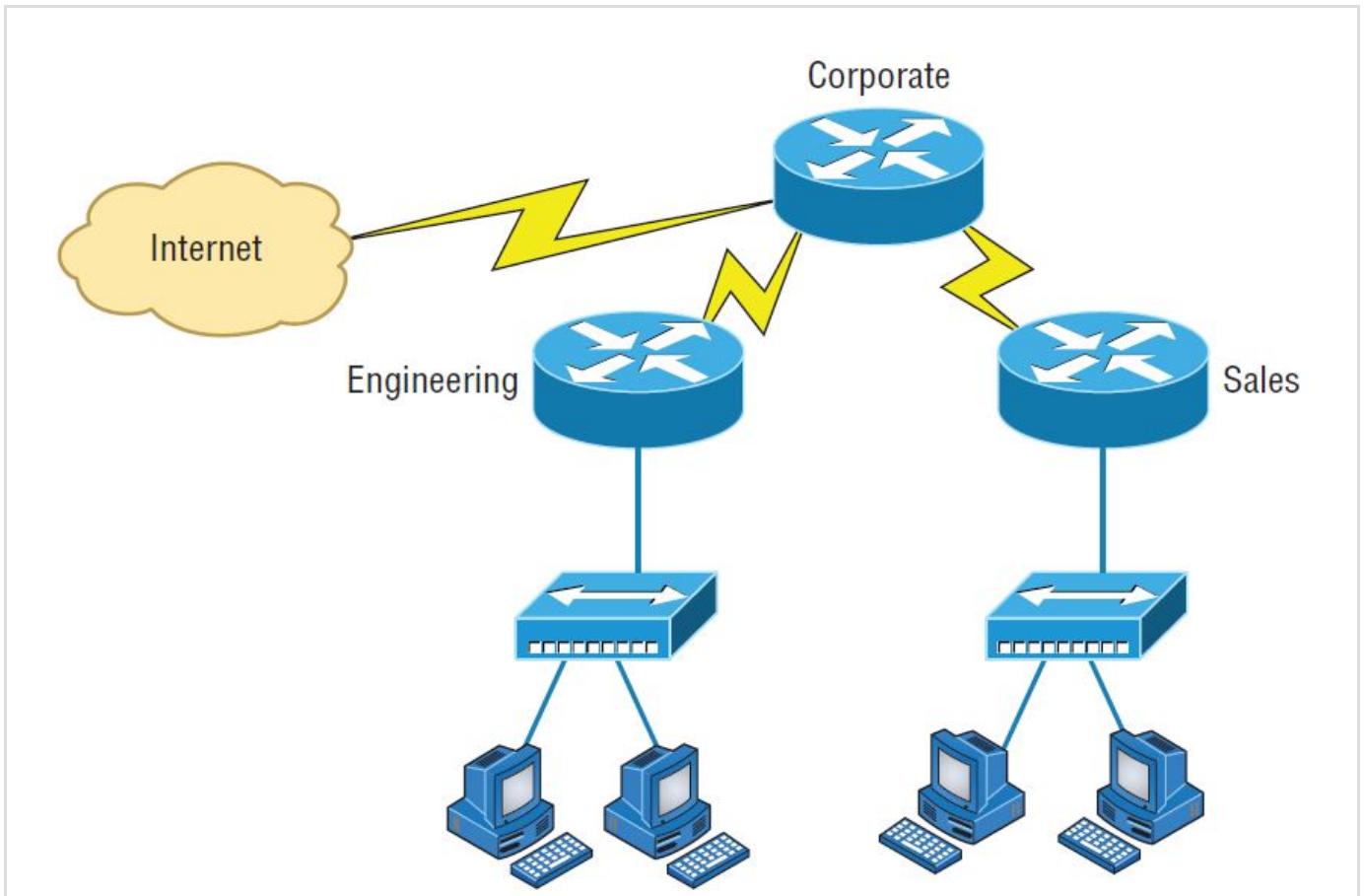
Similar to **Classless Inter-Domain Routing (CIDR)**, the original intention for **NAT** was to *slow the depletion of available IP address space* by allowing many private IP addresses to be represented by some smaller number of public IP addresses. Since then, it's been discovered that NAT is also a useful tool for network migrations and mergers, server load sharing, and creating "virtual servers."

At times, NAT really decreases the overwhelming amount of public IP addresses required in your networking environment. And NAT comes in very handy when two companies that have duplicate internal addressing schemes merge. NAT is also great to have around when an organization changes its ISP and the networking manager doesn't want the hassle of changing the internal address scheme.

Here's a list of situations when it's best to have NAT on your side:

- You need to connect to the Internet and your hosts don't have globally unique IP addresses.
- You change to a new ISP that requires you to renumber your network.
- You need to merge two intranets with duplicate addresses.

You typically use NAT on a border router. See the below figure, where NAT would be configured on the Corporate router.



Where to configure NAT

While NAT has many advantages, it also has disadvantages too.

Advantages	Disadvantages
Conserves legally registered addresses.	Translation introduces switching path delays.
Reduces address overlap occurrences.	Loss of end-to-end IP traceability.
Increases flexibility when connecting to the Internet.	Certain applications will not function with NAT enabled.
Eliminates address renumbering as the network changes.	

Previous

Next

Network Basics

Network Address Translation (NAT)

Types of NAT

- **Static NAT (SNAT)** - This type of NAT is designed to allow one-to-one mapping between local and global addresses. Keep in mind that the static version requires you to have one real Internet IP address for every host on your network.
- **Dynamic NAT (DNAT)** - This version gives you the ability to map an unregistered IP address to a registered IP address from a pool of registered IP addresses. You don't have to statically configure your router to map an inside-to-an-outside address as you would in static NAT, but you do have to have enough real, bona fide IP addresses for everyone who's going to be sending packets to and receiving them from the Internet.

Oops.. No translation found.

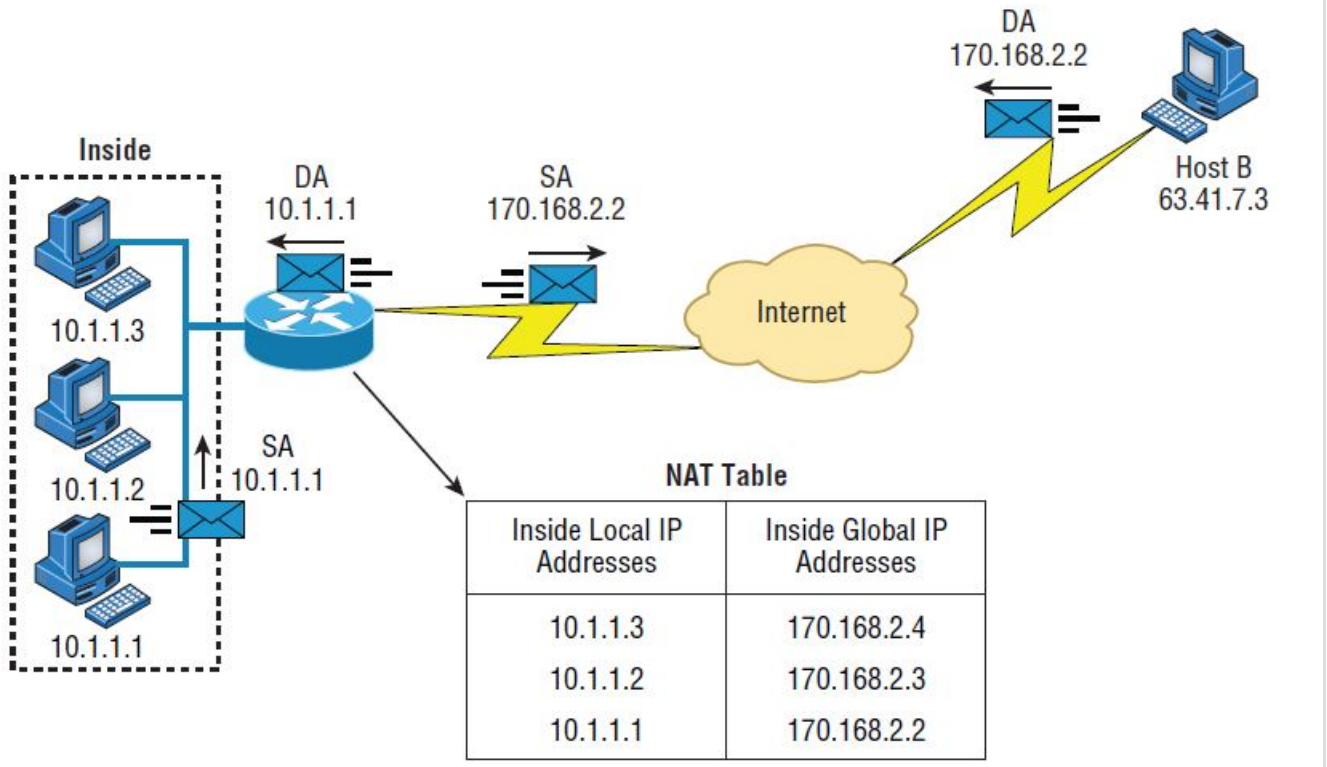
- **Overloading** - This is the most popular type of NAT configuration. Understand that overloading really is a form of dynamic NAT that maps multiple unregistered IP addresses to a single registered IP address—many-to-one—by using different ports. It's also known as **Port Address Translation (PAT)**. And by using PAT (NAT Overload), you get to have thousands of users connect to the Internet using only one real global IP address. NAT Overload is the real reason we haven't run out of valid IP addresses on the Internet.

Previous Next

Network Basics

Network Address Translation (NAT)

How NAT Works

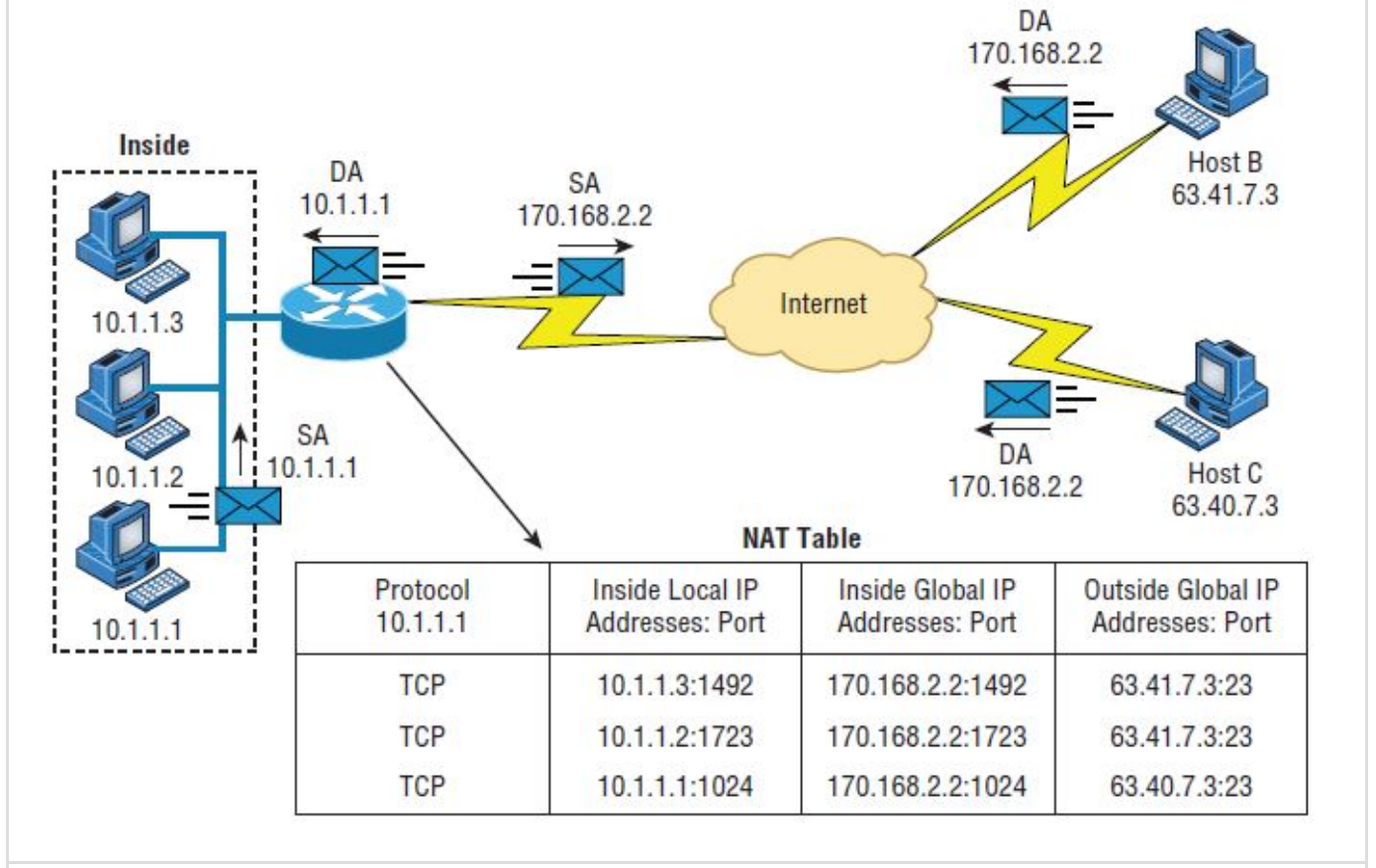


Basic NAT translation

In the above example, host `10.1.1.1` sends an outbound packet to the border router configured with NAT. The router identifies the IP address as an inside local IP address destined for an outside network, translates the address, and documents the translation in the NAT table.

The packet is sent to the outside interface with the new translated source address. The external host returns the packet to the destination host, and the NAT router translates the inside global IP address back to the inside local IP address using the NAT table.

Let’s take a look at a more complex configuration using overloading, or what is also referred to as PAT.



Protocol 10.1.1.1	Inside Local IP Addresses: Port	Inside Global IP Addresses: Port	Outside Global IP Addresses: Port
TCP	10.1.1.3:1492	170.168.2.2:1492	63.41.7.3:23
TCP	10.1.1.2:1723	170.168.2.2:1723	63.41.7.3:23
TCP	10.1.1.1:1024	170.168.2.2:1024	63.40.7.3:23

NAT overloading example (PAT)

With overloading, all inside hosts get translated to one single IP address, hence the term overloading. Again, the reason we have not run out of available IP addresses on the Internet is because of overloading (PAT).

Take a look at the NAT table in the above figure again. In addition to the inside local IP address and outside global IP address, we now have **port** numbers. These port numbers help the router identify which host should receive the return traffic.

Port numbers are used at the **Transport layer** to identify the localhost in this example. If we had to use IP addresses to identify the source hosts, that would be called static NAT, and we would run out of addresses. PAT allows us to use the **Transport layer** to identify the hosts, which in turn allows us to use (theoretically) up to 65,000 hosts with one real IP address.

Using a router or firewall, you can also perform port forwarding, which is translating the port number of a packet to a new destination. The destination may be a predetermined network port (using an IP protocol, but typically TCP or UDP ports) on a host within a private network behind a NAT router. Based on the received port number, a remote host can communicate to servers behind the NAT gateway to the local network.