NAT, DHCP, APIPA, and ARP

Network Address Translation

Network Address Translation (NAT) has become one of the most prevalent networking technologies. It was originally designed to translate addresses from one range of IPs to another, like if a network was using one range of addresses and needs to use another and wanted to avoid reconfiguring everything. However, NAT is mostly used nowadays for hosts on a local network to share an IP address on the internet. NAT is the major reason that the internet still works despite the fact that IPv4 addresses have more or less run out. Eventually, we'll probably all switch over to IPv6 and NAT will be less common, but that seems to be far into the future.

How NAT works Typically on a local network, especially a home Wi-Fi network, there will be several hosts with addresses like 10.0.0.2, 10.0.0.3, etc. and a router with address 10.0.0.1. Those are all local or *internal* addresses. The router also has a global or *external* address, which is an actual address on the internet that it can use to communicate with the rest of the world.

When a host on the local network, say 10.0.0.2, wants to contact an external server like `msmary.edu`, it sends a packet with source IP 10.0.0.2 and destination IP being the IP of `msmary.edu`. There is also a source port on the request, usually a random port like 45678. When the request reaches the router, it replaces the source address 10.0.0.2 with its own global address. When a reply comes in from `msmary.edu`, the destination address will be the router's external IP. How does the router know which internal IP to send it to? It can't just use the source IP (of `msmary.edu`) because maybe multiple hosts inside the network are all making connections to `msmary.edu`.

Instead, it uses the source port. In our example, when the host at 10.0.0.2 sends the packet, it uses source port 45678. The NAT router records this value in a table. It might keep the same port number in the request it sends over to `msmary.edu`, or it might change it. When a response comes back from `msmary.edu`, the router will look at the port on the response, consult its table, and forward the request to the right host on the network. Sometimes people refer to this process as NPAT, with the P standing for "port".

Consequences of NAT NAT makes it so that computers inside the network cannot be reached unless there is an entry in the NAT router's table. If there's not a table entry, then the router would not know where to send the packet, and it would drop it. There are both good and bad aspects to this. The good is that NAT acts a little like a firewall in that it's difficult for outsiders to directly contact a machine inside the network. It also hides the structure of the internal network from the rest of the world. However, it is not a substitute for a firewall, as there are ways around it. The bad is that NAT makes it hard for legitimate people on the outside to make a connection to a computer behind a NAT. This affects people that want to run websites from their home and it affects peer-to-peer connections in video games.

Contacting a site behind a NAT There are various ways around the problem of being able to contact a device behind a NAT. A simple one is called *port forwarding*. This is where you go into your router's configuration to put an entry into the table saying that all traffic at such-and-such port should always be forwarded to a specific internal host. So to run a web server from your house, you could tell the router to send all traffic with destination port 80 to be forwarded to the host on your network acting as the server. A downside is that there can be only one server running at the standard HTTP port. If you wanted a second one, you would need to use a non-standard port, like 8080. Port forwarding is an option if you have access to your router's configuration, but if you are at an organization, then you would have to convince the network administrators to do the port forwarding for you, which might not go over well.

If port forwarding is not an option, there are various *hole-punching* techniques. One is called *Session Traversal Utilities for NAT* (STUN). It involves a third-party server that you contact. When you send a message to that server, it will tell you what IP and port number it sees. These will be the external IP of the NAT router and the port number corresponding to its table entry. Once you have this, you can contact others with this information, and, depending on the type of NAT you have, they may be able to contact you using that IP and port to get through the NAT. If the NAT is *symmetric*, then this won't work. In a symmetric NAT if you contact site A using

port number *p*, the symmetric NAT will only forward things back to you at that port if they came from site A and not if they came from anywhere else.

Sometimes, two hosts A and B, can contact the third-party server. It tells A what B's IP and port are and it tells B A's IP and port. A and B then send messages to each other using that information. Doing so will create entries in each of their tables. The first message each sends to the other won't get through the NAT, but once the table entries are created, subsequent entries may be able to get through.

Another option is *Traversal Using Relays around NAT* (TURN). This also uses a third-party server. Here the server acts as an intermediary or relay between two hosts both behind NAT. Each sends messages to the server who relays them to the other. This works even with a symmetric NAT because the connection always goes through the third-party server and no one else.

Carrier-grade NAT Some ISPs are now using *carrier-grade NAT* (CGN). This uses the IP address range 100.64.0.0/10. This allows the ISP to conserve its own block of real IP addresses. It is basically a second level of NAT on top of the NAT provided by most home routers. This, of course, makes NAT traversal more challenging.

What people think of NAT NAT is a polarizing technology. Many people appreciate that it helps conserve IPv4 addresses, allowing IPv4 to continue to work. Others don't like how it makes ordinary networking things more difficult. In particular, it breaks the *end-to-end principle* that hosts should be in charge of everything and routers should only forward things. People also don't like how it uses a layer 4 concept (port numbers) to do layer 3 work.

Dynamic Host Configuration Protocol

The *Dynamic Host Configuration Protocol* (DHCP) is how you get an IP address when you join a network. In the old days, a network administrator would have to manually assign you an IP address. This is a *static* allocation, and it is a pain on large networks with many different devices constantly hopping on and off the network. In DHCP, there is a server that has a pool of IP addresses that it gives out to devices that want to join the network. Large networks might have multiple DHCP servers running. On your home network, your Wi-Fi router acts as a DHCP server, usually assigning devices on the network addresses in the 192.168.0.0/16 or 10.0.0.0/8 range. DHCP works via the following four-step process.

1. When a device wants to join a network, it sends a *DHCPDiscover* message. That device doesn't yet have an address on the network and it doesn't know the address of the DHCP server either, so it broadcasts the message on 255.255.255.255 and uses 0.0.0.0 as its source address. This message is the device indicating that it wants an IP address on the network. The message also contains certain things the device would like to know about the network, like the subnet mask and location of the DNS resolver.
2. The DHCP server responds with a *DHCP Offer* message. This contains the IP address the server is offering to the device, and it will contain answers to some of the things the device is requesting, like the subnet mask and DNS resolver location. In a large network with multiple DHCP servers, it's possible that more than one of them may send a DHCP Offer.
3. The device responds with a *DCHP Request* message. This is for the device to confirm that it is accepting the server's offer. Any other DHCP servers on the network will see this since it's broadcast, and they will silently withdraw their offers. Actually, before the device responds, it's supposed to check via an ARP query (see below) whether or not the IP address is in use by some other device.
4. The server responds with a *DHCP ACK*. This acknowledges the DHCP Request message. The device now has the address.

Below is a picture from Wireshark showing the whole process.

	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0x8a020b6f
2	0.234390	192.168.1.1	255.255.255.255	DHCP	590	DHCP Offer - Transaction ID 0x8a020b6f
3	0.235406	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request - Transaction ID 0x8a020b6f
4	0.245981	192.168.1.1	255.255.255.255	DHCP	590	DHCP ACK - Transaction ID 0x8a020b6f

Here are a few notes about DHCP.

- Often a device will be rejoining a network, and in the DHCP Discover message, it can request the same address that it had when it was last on the network. If that address has been given out to someone else, the server will send a DHCP NAK (negative acknowledgement).
- When you get an IP address from a DHCP server, you are only *leasing* it. That is, you only get it for a specified amount of time, usually anywhere from a few hours to a few days, though it could theoretically be really short or really long. If the server didn't do this, then it would essentially "lose" IP addresses to devices that aren't coming back to the network. Usually well before a lease is set to expire, a device will try to renew its lease. If it is unable to renew its lease it will eventually try to *rebind*, namely go through the four-step process again. You can use `ipconfig /all` (Windows) and `ifconfig` (Mac/Linux) to see the lease time for your IP address.
- DHCP messages are sent via UDP ports 67 and 68.

Automatic Private IP Addressing

Automatic Private IP Addressing (APIPA), also called *stateless autoconfiguration*, is what a device does when it needs an IP address and can't get one via DHCP or via a static allocation. IP addresses are needed for certain types of communication on a network, like if a device wants to access a web server on a local network or perform a file transfer with another device on the network. Sometimes, a network won't have a DHCP server or a network admin to statically assign addresses, so a device will have to take matters into its own hands.

The address range 169.254.0.0/16 is set aside for this purpose. Just like private address ranges, like 10.0.0.0/8, this is only for local networks, and it's specifically just for APIPA. Addresses in this range are called *link local* addresses. The way it works is the device randomly picks an address in this range. Then it does a probe using ARP (see below) to see if that address is in use. If it's free, then the device gets it, and otherwise it tries again. IPv6 also has this, though things are a bit different.

MAC addresses

IP addresses are layer 3 (network layer) addresses. They are for when you need to send a packet across the internet to another machine. But when it actually comes down to direct computer-to-computer connections, where you have to transfer a packet directly from one computer to the other, then we are at layer 2 (data link layer). At this layer, a separate addressing scheme, called *media access control addresses* (MAC addresses), is used. Historically, layer 2 and layer 3 developed separately, serving different purposes, which is why we have two different types of addresses. A MAC address is only visible to people on the same network as you. They are below Layer 3, which means they are not sent out of the network.

MAC addresses are 48-bits long, usually written as six groups of two hex digits, like AB:CD:EF:12:34:56. Sometimes dashes are used instead of colons. At 48 bits, there are theoretically $2^{48} \approx 280$ trillion possible MAC addresses.

It would be very confusing if two devices on a network had the same MAC, so a scheme is in place to help ensure that every device has a different MAC. The first three groups are a manufacturer code. Device manufacturers can apply to get one of these codes. For instance, C1:C0:C1 is a code belonging to Cisco. Cisco can set MAC addresses for their devices in a range from C1:C0:C1:00:00:00 to C1:C0:C1:FF:FF:FF. This is around 16 million total addresses. They make sure not to repeat any of those on different devices, and they can apply for a new code if they need more. There are various tools online to look up these codes, and tools like Wireshark will automatically show them.

On Windows, to see your MAC address, you can use `getmac` or `ipconfig /all`. On MAC and Linux, `ifconfig` will show it.

Address Resolution Protocol

The *address resolution protocol* (ARP) is used to tie IP addresses to MAC addresses. Here is a common scenario: on a LAN, there is a router with local IP 10.0.0.1 and several hosts on the network. The router is often a NAT router with an external IP on the internet. A packet comes into the router and it needs to be forwarded to the host with IP 10.0.0.2. In order to actually move the bits of the packet over to 10.0.0.2, the router needs to know the MAC address of the host at 10.0.0.2. It uses ARP to get that MAC address.

The router broadcasts a message to the whole local network asking "Who has 10.0.0.2? Tell 10.0.0.1." The host that is using 10.0.0.2 will then respond directly to the router with its MAC address. In order to avoid having to make an ARP query for every single packet, it will cache the result in its *ARP table* for anywhere from a few seconds to a few minutes, typically.

Any machine on the network can make an ARP query. A common example is when a machine needs to know the MAC address of the network's router. Also, when a host needs to check if an address is in use, usually as part of DHCP or APIPA, it will send out an *ARP probe* asking who has that address.

When a host first comes onto a network, or if it switches to a different interface, say from ethernet to Wi-Fi, and its MAC changes, it may send out a *gratuitous ARP* announcing its MAC address. The message gratuitous in that it's an answer to a question that wasn't asked. Anyone on the local network that sees one of these will cache the answer in their own ARP table. On all major operating systems the `arp` command (`arp -a` on Windows) can be used to see your system's ARP tables. You will likely see both static and dynamic entries in there. Dynamic entries were generated via ARP, while static ones are hardcoded in by humans or put there by the operating system. For instance, here is the result I get from the command.

```
Interface: 192.168.1.114 --- 0xd
  Internet Address      Physical Address      Type
  192.168.1.1           c0-c1-c0-88-dc-dc     dynamic
  192.168.1.110         dc-d3-21-b6-0f-fe     dynamic
  192.168.1.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

The first entry came from an ARP query my computer made, asking for the router's MAC address so that I can send stuff to it. The second entry is some other device on my network that sent out a gratuitous ARP. The rest of the entries are configured there by my operating system. Notice that 192.168.1.255 and 255.255.255.255 are both tied to the MAC address ff-ff-ff-ff-ff-ff, which is the broadcast address.