IP Addresses Explained

What is an IP address?

Think of an IP address like you do a mailing address. It's basically the exact same thing. Consider this, when you send out a letter, what do you do? You put that letter in an envelope, write the address, put a stamp on it, stick in the mailbox and put that red flag up on your mailbox, right?

Apply this to computers: that letter is data, the envelope is the TCP/IP header which is actually wrapped around the packet, just like an envelope. The address would be the IP address, located on that header. You stick that in your mailbox, or in the computer world, a buffer, and it gets sent off to the recipient.

All the traveling that a letter goes through, a packet goes through. A letter sent through the mail is taken from your mailbox to your local post office. From there, it's sent to a state post office distribution center. Then it gets sent to a regional center. If it's an international letter goes to one of the national ones. The letter then goes back down this chain until it's in the recipient's mailbox. At each stop, a decision is made on whether or not that that particular distribution center should send it to a higher level, or if it can put that letter in a bin at it's location which will be sent to the next lower level.

Lets say I want to send you a letter, we both live in India but in different cities. I send out the letter, my local post office looks at the address and determines that it can't deliver the letter to the recipient so it sends it to the state post office. The state post office says yes, I can deliver this to the letter bin that belongs to that city and it'll go out with the next truck. Your city post office gets the letter and sees the address, says yeah he's in our area so we can definately send this to him. It gets put on a truck with other letters for people in your area and the driver puts it in your mailbox.

Same thing happens with a packet, except there are no internet truck drivers and it happens a helluva lot quicker. Your packet gets sent to your ISP who routes it back to a local address if it can, or sends it to the next higher ISP until it's on the Internet Backbone. Once there, if the recipient is still in your country, it'll go to his regional ISP, and so on and so forth until he gets it.

Why all this talk about regular mail? Again, an IP address is nothing more, nothing less than an address. What about routers, we have internal IP addresses! Ahh good question, think of an internal IP address the same as having a post office box. The post office gets it and decides to route it to you that way, your physical address obscured to the sender.

How the addressing scheme works?

Okay, now that we've established what an IP Address is I'll show you how addresses are assigned.

IP addresses are broken up into several classes, A to E.

Class A addresses are 1.xxx.xxx.xxx - 126.xxx.xxx.xxx
Class B addresses are 128.XXX.xxx.xxx - 191.XXX.xxx.xxx
Class C addresses are 192.XXX.XXX.xxx - 223.XXX.XXX.xxx
Class D addresses are 224.xxx.xxx.xxx - 239.xxx.xxx.xxx
Class E addresses are 240.xxx.xxx.xxx - 254.xxx.xxx.xxx

Okay, what the hell is all that? I'll explain:

IPv4 address consist of 4 octets. For an A Class address, the first octet, ranging from 1 - 126, describes the network you're sending a packet to. The rest of the octets describe the node. (Kind of like a City, and a Mailbox, respectively.) (A node is just another name for a device connected to the network.)

In a Class B address, the first two octets represent the network and the last two describe the node.
Wait a minute! What happened to 127? Addresses starting with 127 are loopback addresses. That means it points to your computer. Go ahead, ping 127.xxx.xxx.xxx where the xxx is between 1 and 255. It will resolve to 127.0.0.1 and give you a reply that's hopefully less than 1 milisecond.

Anyway, Class C addresses are start with 192, go up to 223, and the first three octets describe the network and the last octet describes the node.

Okay, break for a second. A network with only 254 nodes?!? (0 is used kind of like a wildcard and 255 is broadcast.) That's right, Class C networks are usually given to small time ISP's and organizations who only have a few nodes on their network. Big companies like IBM and the government are given Class A addresses because they're bound to have lots of nodes. When the internet first kicked off, lots companies were told that certain address ranges would be reserved for them. This has caused problems because 1) Those companies aren't using nearly all of those addresses and 2) we're running out of addresses to assign people. The solution for this is IPv6, where we have 6 octets instead of 4, increasing the number of addresses exponentially.

Alrighty, back to classes. Class D addresses are for multicasting and Class E addresses are reserved for future use.

Ports

If I had a dime for every time I've heard, "I scanned xyz and found this many open ports!! How do I exploit them?".

Ports are logical addresses on a computer where applications send and receive data. That's it, just like IP addresses, ports are nothing more and nothing less than addresses. To put it in the same perspective as regular mail, think of your computer as an office building and applications on your computer as different companies within that office building. Now your office building isn't going to have different mailrooms for each company. They'll receive mail and put it in your company's bin and you come and get it. Your computer has 66,536 some odd ports, plenty for all of your applications, so your office building has some 66,536 odd mail bins for companies in your building. Will you use all of these ports? Most definately not.

So you scanned a computer and lets say it has 100 ports open. Great, that just means that your "office building" has 100 mail bins, it doesn't mean that there are companies in your "office building" who use those mail bins. You have to have some kind of application running that relies on a service that uses that port in order to do ANYTHING with it. Just because you found xyz port open, that doesn't mean that you're getting instant access to their computer. If there's no service listening to that port, you're doing the same thing as sending letters to a mailbox that no one checks.

Also, think of a Firewall as a person in the basement mailroom analyzing each letter to determine whether it's junkmail or something that a company doesn't want. Either that or he can hold all mail for that company until instructed otherwise.

Exploiting Services

I'm going to touch briefly on this subject for two reasons. 1) I'm not the most knowledgeable person on the subject and 2) It's illegal unless you own the computer you're exploiting or have the expressed permission of it's owner.

Okay in order to exploit a service, you first have to find a vulnerability. That isn't always easy because the remote computer more than likely isn't going to just give you the source code for the service running on a port you want to attack. So if you're wanting to find vulnerabilities in their service, you're going to have to guesstimate what kind of service is running on that port and get your own and analyze it's code. There's other methods of testing for vulnerabilities. Most well known vulnerabilities (read: ones you find on the internet) are patched already. You can test these and see if they're patched, but chances are that if the admin on the other side is even a little vigilant, he'll keep his software updated.

Okay, so he is vigilant and all known vulnerabilities are patched, what now? You poke around and see if you can find one of your own. This varies from service to service and application to application so I'm not going to go into it. (That and I don't know that much about the subject myself.)

Once you find a vulnerability, you'll need to exploit it. Depending on what kind of service it is, this is usually going to involve some code writing. And more often than not, you're going to have to do it their way, meaning write code that the remote service will understand. Again, this varies from service to service and application to application.

Packets

Okay, now that we know a little bit about an IP address, I'm going to go a little more into depth. If having an IP address is like having a mailbox, then it'd be pretty useless unless we sent or received letters, right? That's basically what packets are.

A packet can be broken up into a few sections. The first section is the header, which contains a bunch of info. The most obvious is the destination IP address, duh. Some others are version, source address, TTL, type, and others.

Some are obvious, some aren't. TTL is an abbreviation of Time To Live. It's a value in milliseconds describing how long that packet has to live before it "dies" or gets discarded. The thing with the internet is, it's not extremely reliable. You get disconnects, reroutes and all kinds of crazy stuff that happens. If packets didn't have a TTL, we'd have a crapload of lost packets floating around aimlessly looking for a host which could very well have found its way to a dumpster, especially if it was made by Dell. The best way we've found to avoid these lost packets is to have them commit suicide if they can't find their destination. Kinda depressing, huh?

Anyway, there's about as many different types of packets as there are protocols, which is a bunch so I'm not going to list many. Some are: TCP, UDP, IMCP, SCTP, RTP, etc.

There's an ID number for each packet. That's so the host can know how to rearrange them so that the application using that data can understand it. One thing I almost forgot to explain. When you send data over the wire, it's not exactly streamed. It's broken up into chunks (packets), so that in the likely event that one of those packets becomes corrupted, your computer resends that one packet instead of the whole file again. Makes sense, right?

Alright, then there's the body or payload of the packet. Guess what goes in there? If you guessed data you're right! That's not all that can go in there though. With error checking, one of the things checked is the size of the packet. Now if the last packet only has 10 bytes of data in it, and it's sent like that, the host looks at it, says "That's a lot smaller than the others, something's wrong, send it again" to your computer. So in addition to data, you have what's called 'padding' to fill the void and make the packet the appropriate size.

The trailer (or footer, depending on how you look at it) contains some error checking information and signifies the end of the packet.

Types of packets

Okay, there's bunches of these but I'm only going to list a few, when I start getting paid to write tutorials, I'll quit being so effing lazy, alright?

TCP, probably the most common type of packet out there. It stands for Transmission Control Protocol and is your basic payload packet with error checking. Want more? Go buy a 2 inch book on it, I've only got so much time before I fall asleep typing.

UDP, User Datagram Protocol. It's kinda the same as TCP but without error checking. Now why on earth would I want a protocol that doesn't use error checking?!?!? You may ask. Cool, I'll tell ya in two words, then explain: STREAMING MEDIA. If you miss a frame or two from a streaming video, you probably won't even notice, right? UDP is widely used in online games, media streaming and other applications where all the data doesn't have to be uncorrupted all the time, if you can think of any. Ever play a game and get that annoying "There is a problem with your connection...." message? Yeah and then your helicopter suddenly appears upside down and an inch from the ground? Yeah I hate that too but that's because you're losing a lot of UDP packets for whatever reason.

ICMP otherwise known as the Internet Control Message Protocol. This is how your ping application works. That's not its only use though. It's used to send error messages, timestamps, and address masks.

RTP, Realtime Transport Protocol. Alright, all I knew about this was it was like UDP in the aspect that it deals with media. Come to find out, it deals largely with VOIP (Voice Over IP) and was originally designed for multicasting. Gotta love Wikipedia.

Packet switching and Circuit Switching

Briefly, packet switching is basically using multiple routes and let the packets find their own way. This is your normal flavor of internet. This way, if one of those routes closes due to a sledge being taken to a stubborn server, the packets just take another route.

Circuit switching is when your computer finds what is supposidly the best path to a destination and sends the packets down that route. ATM (Asynchronous Transfer Mode) uses this by sending really tiny packets and it's supposed to be really really fast. The obvious downsides are that 1) your computer has to find the route first and 2) you know what happens when that route closes.

Traceroute

Speaking of routes, there's a program on most PC's called tracert which determines how many hops a packet takes to get from your computer to it's destination. A "hop" is when your packet goes from one server to another. The thing with hops is, they slow you down. RIP, I think, is the routing protocol that only allows 15 hops or something. I degress, the more hops, the more processing that has to be done forwarding your packet and that's why you slow down.

So anyway, traceroute tells you how many hops it takes to get to a destination and it can even list the servers on which your packets hop. It works by incrimenting a packet's TTL by one second each time it hits a server. When a server receives a traceroute packet with a TTL of one second, it sends an ICMP packet back, letting traceroute know the who/when/where.

For more info on Traceroute, visit http://lwn.net/Articles/89597/.

References:

http://www.ralphb.net/IPSubnet/ipaddr.html
http://www.wikipedia.org
http://lwn.net/Articles/89597/

Source: http://www.go4expert.com/articles/ip-addresses-explained-t2445/