etworking for the masses
By sam sn33d

Overview-

What this is about:
- Networking
- Proxies and Gateways
- Routing Methods
- Software and Hardware for Networks

Introduction

First off, I think that it's important for everyone to understand networking protocols and the ways that they are run. Many security problems arise in networks, and if you hope to understand those attacks, you need to have a basic knowlage of how this works. Back when the internet was first being designed, John Anderson and his team of people had designed several sketches that were going to be the lay out for the network they called the ARPAnet. This project paved the way for many of the lan structure's out there today, such as the model we're going to talk about now. I showed this to a friend of mine at school, and he asked me some questions about what I wrote, so I'll answer your questions in the next issue. I plan on doing a networking FAQ, but if people don't need one I see no need to write about this topic anymore.

The hub-router-server model

This network (a LAN in this case) model was designed for buisnesses, and was actually used by the government at one time. It was spawned off some of the sketches done earlier by Johnson. What happens here, is this: There are several work stations, and most in a group (say, a room) are connected to a hub. Many hubs are then connected at the router, and then many routers are connected to a server. Now, in most instances, all of these are hard wired for reliablility reasons, and are connected at any given time that the server is up (usually 24.7). Most of

the time, or at least from what I've seen at schools and other networks, they are connected using co-axial cable. Data can be sent from work station to work station, and if the information must travel beyond hub, the packets go to the router and determine where to go according to the router table. As in some networks, there are hundreds of workstations, and routing software can soon build up a large table which will in some cases not be entirly correct at a given second. The networking software has to determine how long of a period is necissary before sending packets to each link to determine the new network topology. A new network state can't be updated every nanosecond, however it must keep a current enough version to rely on the table and assume that data will not be lost in the scheme of things. There are a few ways to set up network tables besides bombarding the router with updates. One variation of this is Distance Vector Routing. This is a list (not a table mind you,) but a set of data set up in this fashion: [destination place, jumps]. The jumps are calculated by adding the number of jumps (or router transfer points) and then placing that number in the second slot. The first is often just a name or an assigned variable. These are not usually sent to every machine possible on the network, but just the one jump router path boxes. Neighboring routers compute the paired information, and then draw out their own topology map. The other routers can figure out the number of jumps by sending the relaying info to a closer machine, then figuring out if it can make it in one jump. So if machine E makes it's list, according to this:

D------E------F

It would distrible a list similar to [D,1] [F,1] (for node E) and send it to the neighboring machines and the router that it is connected with. D would send [E,1] [F,2]. There is a huge problem here that we will get into later down the paper. Then each computer (D,E,F in this case), can make their own tables according to the [D,1] (or whatever) packets of information they recieve. This takes the load off of a main router table, evening out all the data through the network. You can see if link E-F breaks, then D still assumes the it can do a 2 jump to F, not knowing that link E-F is down because it hasn't recieved an update from E yet. E knows that the link is down, and would ask the neighboring routers for a jump cost to get to F. E would try to route the data through D, because D is still telling all the routers that it can 2 jump to F. This is called the count to infinity problem. Solutions to this problem have now been worked into most software, as it can result in _lots_ of lost information. Routing for bigger networks uses Link State Routing (LSR) and sends out a 3 section list like this:[Router path, Destination,

number of jumps]. This is for larger (and I mean LARGER) networks. Someone would implement this if they had a class A or B network. These triplets are sent to the neighboring router, and from the information they recieve the router can make an accurate up to date model of the network.

(***NOTE*** This doesn't go into DVR very much, so if you think you'd like to know more about this protocol, I have a fine list of books to read. That goes for any of the information typed here.)

Sharing IP's and Getting different addresses-

__________ ____ __________

| Cable |_______|Hub |_________|Computer 1|
|__________| |____| |__________|
|
________ |_________
| Computer 2 |
|_________________|

As more and more homes are buying two or three computers to work with, and more and more of these people are starting to use cable connections. The most efficient way to network these computers together would be to buy a hub (usually $50 or so) and connect however many computers to the hub. Now, between the hub and the cable connection, you would most likely need to use Network Address Translation (NAT). (***SIDE NOTE*** If you plan to do this, buy seperate IP addresses for each of the computers. It's usually not good to be sharing an IP address in this fashion here, the router will have a hard time.)

@ Home offers seperate IPs for each computer around 5 bucks) NAT will be able to decipher the data sent between the computers and retrieve the information through the cable. Another way to do this is to set up a linux box as a proxy (which would actually take the place of the hub in the drawing above), and do all your data transfer through the box. The ISP never knows about the computers behind the proxy, it just sees the data going to and from it. Cable modems are great for speed and data transfer, but watch for sniffers laying about on your network. Also if you're like this, try getting a packet sniffer on the main cable network so it's linked through everyone else in your area. (<disgression> Since this paper is _not_ about sniffers, I won't go into this very hardcore. Some people asked me to explain sniffers better, and here it is. Sniffers are small, usually

unnoticed background programs that monitor network traffic and packets, searching for passwords and other sensitive information. It takes this data and stores it in a temporary file, with a very long file path. Because the deeper directories are less traveled, there's less of a chance that someone will stumble across the goods. If you want some names, use nitwit.c or tcpmon.c to do this. It depends on what type of protocol the ISP uses too. </disgress>) So you can see, networks aren't very secure, and now the ISP's have technology to monitor your every move through the networks (whether that be TCP or UDP packets leaving and entering), and I'm a little disgruntled at this. However, that's not my point. I disgress again :P.

Multicast-

This is a little different than what has been discussed so far. Let's make an example. Node A would like to send info to node A1 to A100. Since many of these requests at the same time may bog down a network, multicast is used. Nodes A1-100 make a type of "subscription" to a router, and node A would send the data to a router using multicast. Then the router tells A1-100 that there's data waiting for them, and if all of the recieving nodes are subscribed to get data from node A, then the data is transferred to A1-100. Multicast isn't very sucessful for a network that needs say, class C or B IP addressing (***NOTE*** 152.x.x.x is class A addressing. 209.123.x.x is class B. 123.123.123.x is class C. 209.12.23.34 is class D routing.) , so the multicast group is usually static on a class D network. This saves routing table space, because the sender (A), doesn't need to know the route to get to the recieving nodes (A1-100). In fact, they don't need to know anything except the multicast router and the identifer for each node. In unicast, which is all above this paragraph, things are recieved and then sent, whereas in multicast, packets are duplicated for hosts and sub-nodes.

protocol index-

BGP- Border Gateway Protocol, this is used for routers to talk to each other, and establish a topology of the next-door-neighbor networks. Using an algorithm approach, it connects to neighboring routers and gets info for the next router, and the next, and so on. Note: This uses TCP connections to talk to each other

preventing loss of data. More protocols will be discussed in a different article in a future issue if people want me to write.

[/sam sn33d]

[sn33d@hotmail.com]
[samster083----AIM]
[29044813----ICQ]

I will try to get another networking paper together for happle 10, as there are more protocols I'd like to explain. Please give me feedback on this, because I don't want to do another paper similar to this one if no one sees a need for it.