The following paper was originally published in the
Proceedings of the Eleventh Systems Administration Conference (LISA '97)
San Diego, California, October 1997

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org

Creating a Network for Lucent Bell Labs Research South

Tom Limoncelli, Tom Reingold, Ravi Narayan, Ralph Loura
– Lucent Bell Labs

ABSTRACT

This paper describes the tools and techniques used to split the AT&T Bell Labs Research networks in Holmdel and Crawford Hill into separate networks for Lucent Bell Labs and AT&T Labs as part of the ''tri-vestiture'' of AT&T in 1996. The environment did not permit us to keep the system down for an extended period of time. Legacy systems and old configurations were supported while new systems were introduced. We did not have direct control over many machines on our network. This paper describes the old network and what we were trying to build (or split), but focuses mostly on the specific techniques we used or developed. What made our network unique is the amount of self-administered machines and networks in our environment. We took unmanaged chaos and created two clean networks. This paper is from the perspective of the Lucent Bell Labs system administrators (SAs), not the AT&T Labs SAs. The transition did not go smoothly, and if we could have read this paper before we began we would have avoided many of the problems.

The beauty of it all is that we did not take one mess and create two separate ones, we split and cleansed the networks at the same time.

The Increasing Trends

There is little literature [RFC1916] on the topic of merging, splitting, and renumbering computer networks because when it is done it is often not thought worthy of documenting. We initially did not plan to document our overhaul because we felt we were ''the only ones''[1]. However soon we realized this was not the case for many reasons. [RFC1900] [RFC2071] [Lear1996].

Old networks need to be cleaned. We see a growing trend in research and academic environments where networks have grown organically and ultimately reach a point where they must be pruned and weeded. Networks that grow this way often do so because they are renegades that exist outside of any authoritarian central control. In recent years corporate CIO departments have down-sized from mainframes to Unix environments. Now ''the suits'' use the same computers and protocols that were once the domain of the renegades. Soon centralization or at least standardization becomes an obvious way to improve performance, stability, and, of course, to save money. It can also happen for political reasons. These situations can all trigger the merging of networks.

The Reagan Era was marked by constant corporate buyouts. Now some monolithic companies are splitting themselves. We feel that corporate divestitures such as AT&T's may be a growing trend. The Federal Trade Commission is currently investigating at least one large software company.

[1] at least an insignificant minority.

It was tempting to take one messy network and create two messy networks. We avoided the temptation and instead viewed the corporate split as an opportunity to base-line our systems and generate two clean, well-engineered, networks.

The Old Way

From informal surveys, we found that most sites perform network merges and splits in very simple and unsophisticated ways:

1. Declare a week, weekend, or evening to be ''downtime'' and convert every machine at once.
2. Move one machine at a time.

We preferred the first option, but management wouldn't approve a simultaneous vacation of approximately 500 researchers. Obviously the first option could have been a disaster if we made the same change to every machine over a weekend and then discovered (presumably on Monday morning) that the changes we made were wrong. The second option was too labor intensive. It involves a lot of footwork as a personal appointment must be made with each user. In a chaotic environment keeping appointments can be difficult.

We adopted a hybrid technique.

The Task

On September 20, 1995, AT&T announced that it would separate into three companies: AT&T, which would retain the Long Distance and other Services businesses; Lucent, which would retain the telephony

and data systems and technology business; and NCR, which would retain the computer business that they had before they were acquired. Our user base and network had consisted of the computers and networks related to AT&T Bell Labs Research in Holmdel, NJ and the Crawford Hill Laboratory, also in Holmdel, NJ. Thanks to fiber optics,[2] the networks were tightly coupled even though they were three miles apart.

Unlike some divisions that were targeted entirely to one company or another, we were split approximately 40% for AT&T, the new ''AT&T Labs,'' and 60% for Lucent, the new ''Bell Labs.'' None of our users was targeted for NCR.

Before the split was announced we had planned massive changes. Luckily we hadn't yet deployed them. Our plan was to take our 40-odd haphazardly grown networks (consisting of 600 computers) and replace them with one large, switched ethernet network per building, with a number of small subnets for special purpose work.

With the announcement of the company split, we had to adapt our plans to the new scenerio. Luckily we had already done the homework required. We essentially continued with our plan, but did it once for each company. That is, two major switched ethernet networks per building (one for each company) and a dozen or so small subnets for special purpose work.

We only had to split our networks to the point where we had two, completely independent connections to the corporate backbones. AT&T and Lucent would deal with splitting the backbone. They would even relay traffic for approximately six months, though we could use filters on our own routers to simulate the final split to ''check our work.''

Why Our Network Had Grown Organically

Our networks were as organized as a swamp. Originally each small department had self-administered its networks. Departments had been reorganized, system administration had been centralized, re-centralized, gained management support and lost it a couple of times. Many users maintained their own machines for historical reasons. In some groups, the SAs were considered ''secretarial'' and were only expected to create accounts, hand out IP addresses and do backups. The combined network we inherited was therefore quite chaotic. For example, when a researcher felt the network was too overloaded, he would add a second ethernet port to his workstation and create a small subnet for himself. This worked because we used RIP at the time. One of the SA teams even kept /usr/local world-writable so users could install new software as they wanted!

[2]Key elements of fiber optics were invented at the Crawford Hill lab.

What Our Network Looked Like

The network consisted of four main user communities each with its own SA procedures and standards. Although we had recently centralized SA functions, the four communities had never been properly merged. But at least by then, we had eliminated 90% of the UID conflicts. There were four NIS domains, two different automount configurations, and each group had a different /usr/local. Each community had from two to five main networks plus many subnets for experiments or to pacify researchers that felt they were important enough to warrant private networks. As a result, we had approximately 40 Class C-sized networks. We had recently been allocated a Class B network which we had planned on dividing into subnets and transitioning to but this migration had only just begun. Almost all of our machines were connected via 10base-T wiring which was administered by our telecom department. There were also 2-3 pockets of thinnet which were maintained by the local users and were being replaced by telecom's 10base-T wiring as part of this project.

Before the split was announced, we wanted to build 2 major networks for Bell Labs, one in each building, with small subnets for experiments only. We wanted no more private networks than needed. We wanted all networks to be connected to our routers, not hung off a Unix host with two ethernet interfaces. The major networks would consist of ethernet switches tied together with FDDI or some other high-speed network technology. We wanted one NIS domain, one /usr/local (or equivalent), one procedure for everything. When the split was announced, we took our plans and doubled everything: one for each company. We would move machines to the correct NIS domain, network, cut the networks, and be done.

NIS Domains and Stranger Things

Changing the IP address of a machine affects all the machines that depend on it for services (i.e., the clients have to be rebooted after the server is changed). NIS and DNS have IP addresses hardcoded making the change even more difficult. Rather than merge three of the NIS domains into an existing fourth, we created a new NIS domain and merged all four into this new domain. We had to reconfigure more clients this way, but it permitted us to start with a clean NIS configuration that we could experiment with, install security-enhancement tools, and develop modern update procedures and more. It also let us start with a fast machine. Since upgrading an NIS master server is difficult, it's best to use the fastest machine you have on hand, deferring the next difficult upgrade. We were also unsure of what patches and security problems [Hess1992] might have existed on the old master. ([Wietse1996] and other tools help make make NIS more secure.)

Most NIS masters read their source files directly from where Unix usually puts them (i.e., passwd is in /etc). We instead put all NIS databases in /var/yp/master (which we aliased to $M) and used `xed` to edit the files. The advantages are:

- $M/passwd contains the global passwd file while /etc/passwd only contains a minimal password file. Break-ins that steal the NIS passwd table are useless for breaking into the master.
- The /etc/passwd file on the master is very small. It contains only the accounts required (system administrators) and the passwords for these accounts are different than the ones in the global NIS passwd table.
- There is no need for `root` (or any UID 0 account) to be in the NIS passwd table. All clients retain a local root password. If the NIS passwd table is stolen and the root password is cracked, the intruder does not have the keys to all the workstations.

"`xed`" is a script that automates file locking and RCS (to record revision history), and then runs $EDITOR to edit the file. After any file is edited, "`ypmake'`" (our own script) is run manually to perform 100% of the update processing. "`ypmake`" locks a file and runs the Makefile. The benefits are:

- Can't step on people's toes: Multiple people editing the same file or running "make" at the same time had caused problems in the past.
- Reduced training time and confusion: The old systems had each administrative file on a different directory on possibly a different machine. Training consisted more of "where is this file and what do I do after I edit it" than what we consider "real work." Now, no matter what you needed to change the procedure was "`xed $M/foo`" followed by "`ypmake`."
- Since `xed` maintains a RCS log of all changes, we can revert to an old revision of any file at any time. This has proved useful when an error crept into a file, we could use the RCS log to discover when the mistake was made (how long was that service misconfigured) and who made the mistake (so they can be re-educated).

Deciding what to put in $M was easy: We put everything we could think of and haven't regretted it. In fact, some non-NIS files are maintained there, including the configuration file that drives our DNS zone generation system. When the DNS configuration files changed, the h2n program from [Albitz1996] is run automatically and the daemon is signalled to load the new configuration. We put other files into $M, such as our host inventory, list of mail servers, and files related to maintaining our Network Appliance file servers.

The makefile encapsulates all the update procedures in one place. This makes the question "where do I add a new feature" moot. The entire system becomes easier to debug.

Rather than maintain duplicate sets of procedures during the transition, we used the new NIS master to drive the old legacy systems. That is, we used the Makefile to encapsulate the confusing procedures of the legacy NIS masters. The legacy systems had scripts to process each kind of updated file and often the script that had to be run had a different name and location on each legacy system. The Makefile "did the right thing." For example, if a change was made to $M/auto_home, the Makefile would copy it to the correct place on the old NIS masters (and even do some translation for one of them) then run the appropriate update scripts on those masters. Eventually this new master was driving all the tables of the old masters. As the old masters lost their clients, we removed the "push to legacy" portions of the Makefile. (See Listing 1)

NIS slaves were configured to serve the NIS databases of the old and new domains at the same time. Many people do not realize that an NIS slave can serve the data of multiple NIS domains at once. [Stern1991] explains why this works and how to configure a slave to do so. Once this was complete, individual clients could be converted to the new NIS domain "at will" since both domains were available on every subnet. This enabled us to convert a small number of machines for testing. It was critical that the new NIS domain served the proper information before "real" users were exposed to them.

Sidenote: When writing the scripts that automated our processes we adopted a new guideline that reversed a previous tradition: Only write the code for the features you will immediately use. Our older scripts (the ones obviously not written by us) were convoluted and difficult to use because most of the code dealt with features the author thought we might use but didn't. This new guideline reduced our maintenance and simplified everything.

Building The Perfect Pair(s)

We learned the hard way that is is better to wait and deploy all changes at once than to deploy each change globally when we thought it was ready. When the new DNS servers were ready we spent two days making sure that /etc/resolv.conf on all machines pointed to the new servers. However, then we realized that one of the servers would have its IP address changed soon. Since /etc/resolv.conf contains IP addresses, not host names, we had to repeat this global change. We couldn't risk this in the future.

We decided instead to make one perfect SunOS client and one perfect SunOS server and move a couple "friendly users" to these machines for testing. Once this was rolling we created one perfect Solaris client and Solaris server.

Once the needed changes were documented, they could be embodied into a script and our 30 or so

changes could be done at once, and we'd know they were all correct. Every time we had to visit a machine to make a change it is a bother to our users. The worst thing we could do would be to visit each machine 30 times to make 30 changes. With this script, we could make all changes in one (virtual or physical) visit and a single reboot.

We brought our users into this process by placing some of these ''perfect'' machines in public spaces and asked numerous users to log into them to make sure their startup files (.profile, .cshrc, etc.) operated properly. Also, users that were paranoid that something (homegrown tools, etc.) would break in the new configuration were walked to a perfect machine for ''real time'' debugging and feedback. We did find bugs thanks to these users.

As the configurations stabilized, we cloned everything for AT&T Labs. We made the ''one perfect client'' for AT&T Labs, built a server, and modified the script to do the right things if it was run on an AT&T machine.

We also set up a clone of our NIS master for AT&T and gave them our new procedures and automation. In fact, until certain files were split, our NIS master pushed to the AT&T NIS master the same way it pushed other data to the legacy NIS masters.

Our ''reconfigure'' script took two arguments: the new IP addr and hostname; from there it could determine everything else (name of NIS domain, default route, etc. Since the IP address indicated which company the machine was targeted for, even special things needed for particular companies were automatically done.

The ''reconfigure'' script could handle just about any situation but had to be manually brought to the machine. We did this because becoming root on a machine is different in each legacy area. Some machines accepted rcp/rsh as root from some ''master'' server. Other machines did not permit us in as root in any of our usual ways and not all machines could be NFS clients. Therefore to execute it you FTP'd a tar file from a central machine, untared the file in /tmp and ran one script (which asked for an IP address and new host name and absolutely nothing else). The tar file contained the right files for AT&T and Lucent, and the script could make all of its decisions from the IP address it was given, including what company's configuration was needed.

We tested the script on machine after machine until it was bug free for every combination of OS – SunOS or Solaris – and company – AT&T or Lucent. The script was hacked to automate some of the changes for less popular machines, like our microvaxen that certain users just refused to let go of.

Dividing The Fileservers

Some file servers had data of users from ''the wrong company'' that had to be moved. Many file servers were old and the data was moved to new file servers purchased as a result of the tri-vestiture. It was easier to purchase new fileservers than to split old ones. Unlike [Remy1994] we had money to burn, but not people power.

We used netgroups to help our split. Netgroups is a difficult file to edit, and our old system generated one huge netgroup from our /etc/hosts file. Our new system was driven by a meta file ($M/netgroups.config) which ''mk.netgroup'' (See Listing 2) converted to a proper NIS netgroup file. (Our master ''`ypmake`'' Makefile did the conversion automatically, of course.)

The format looks like this:

```
allnfs: +att +bl +unknown
machineA: att
machineB: bl
machineC: unknown
machineD: att bl
```

This would generate netgroups for the att machines, the bl machines, and the unknown machines. It would also generate a netgroup called ''allnfs'' which contained the att, bl, and unknown netgroups. Initially all machines were listed in the unknown group and the fileservers were exporting to ''allnfs.'' As we learned which machines were going to which company, we changed their netgroups. As we eliminated, for example, AT&T files from Lucent file servers, we changed the exports file on the server to export to ''bl.'' The NIS master's Makefile also generated a web page from this file that detailed which machines were in which netgroup so that users could verify our data. This was important because our information about which machine went to which company was spotty and involving users in the process helped dramatically.

We would know this phase was done when all the files were moved to the right servers, all partitions were exported to either ''bl'' or ''att'' but not ''allnfs,'' all partitions on a file servers were exported to the same company, and all the machines were classified with a single netgroup. These tasks could be divided over many SAs and done in parallel.

Merge In

Our plan was to run our main 10 subnets and the two new subnets on an array of ethernet switches. Machines configured for any of these 12 IP subnets could be plugged into this network. We would be able to renumber a machine without having to wait for our telecom department to change the machine's jack to be on their new network.

Multiple IP Subnets On One Wire

It is possible to run more than one IP subnet on an ethernet. This works for the same reason that the same ethernet can have machines that talk TCP/IP, Ethertalk (Appletalk over ethernet), and DECNET connected at the same time. Each computer is smart enough to ignore the packets for the other machines. Since this concept is not commonly known, we will explain how it works.

In Figure 1 we see two workstations connected to an ethernet hub as one would expect. They are on subnet 1.1.1.* (netmask of 255.255.255.0). On an ethernet, all hosts see each other's packets but are smart enough to ignore packets not destined for themselves.

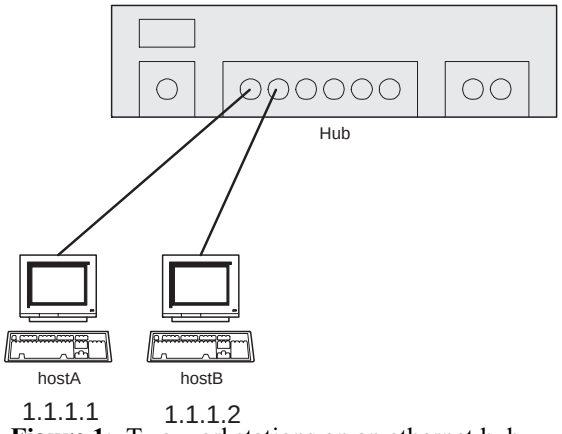

Figure 1: Two workstations on an ethernet hub.

If two machines configured for subnet 2.2.2.* (same netmask) were connected to this hub (see Figure 2), they communicate with each other as one would expect but because the way IP is designed, X and Y would ignore the packets sent between A and B and vice versa. However, host A, B, X and Y share the bandwidth available. That is, while there are two IP subnets, the network is still a shared 10M of bandwidth. That is, while there are two IP subnets on this ethernet, the total bandwidth does not double.

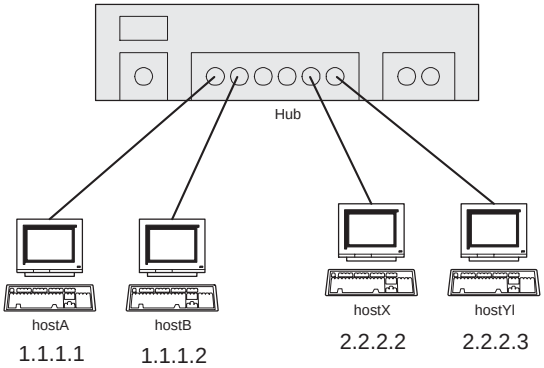

Figure 2: Two subnets on the same hub

Suppose host A wanted to communicate with host X. Normally these two machines would not be on the same hub and host A would send its packet to a router. This router would then deposit the packet on host X's ethernet. In this case, however, host A, X, and the router are all on one ethernet (see Figure 3). There is no magic. Packets still must be sent through a router. In this case, host A will send the packet to the router, who will forward the packet out the same interface it came from (with proper packet header changes, etc.) so that X receives this packet.

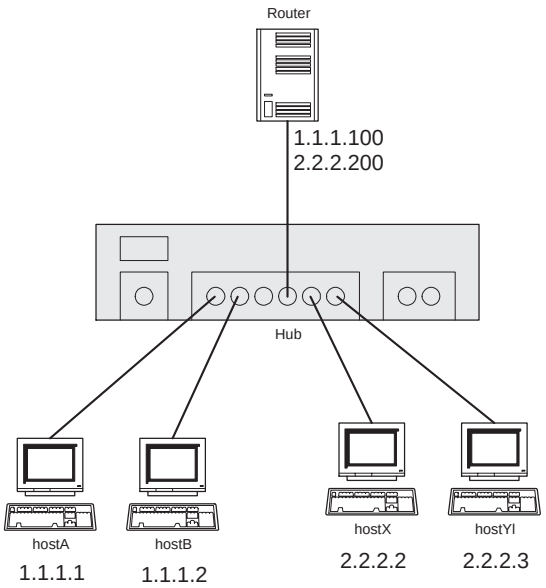

Figure 3: Routing among subnets.

For this to work, the router must be able to assign two IP addresses to the same interface. This is called a "secondary IP address." In our example, host A sent the packet to its default route, 1.1.1.100, and host X received the packet from the router at IP address 2.2.2.200.

The bandwidth used on this ethernet is twice as much as a normal packet since it crossed the ethernet once to get to the router and a second time to get to host X.

Renumbering NFS Servers Made Easy

Changing the IP address of a fileserver means rebooting all of its NFS clients since NFS clients never check to see if a server's IP address has changed. This is because an NFS mount is to an IP address, not a hostname. If we renumbered a NFS server, the clients would hang until they were rebooted, waiting for the NFS server to reappear at the old address.

Unix workstations and servers can be configured to have secondary IP addresses the same as routers. At the time of our split, most of our file servers ran SunOS 4.1.x which does not support secondary IP addresses (Solaris 2.x and IRIX 6.x do[3]). We cleared

[3]See the *ifconfig*(1M) man page

this roadblock by using `vif`[4], a device driver that lets a SunOS machine appear on two IP addresses at once.

We updated our configurations so that all new NFS mounts would be addressed to the server's new IP address. As clients rebooted (for whatever reason), we would eventually have all machines talking to the new IP address. However, now most machines were talking to their main file server through their router. A small price to pay for the ability to renumber clients in a lazy, as needed, fashion. Once all clients had been rebooted, the NFS server is reconfigured to only use the new IP address.

Split Out

The physical split was designed to be neat and orderly after a big merge. To split the networks, we planned on merging the pre-existing production networks into one major network in each building. Since all network jacks could simultaneously support machines of both old and new IP addresses, we could renumber machines at will. Once they were renumbered, we would move machines to the ethernet hub of the appropriate company.

Once the hubs were segregated, we would move the hubs onto the ethernet switches of the appropriate company. Then we could rearrange the connections between the ethernet switches so that they were only connected at once place. Then we would plug the router into the ethernet switches at two places, one for each company.

As with each step of this project, we found there was a fast way to do something that would require downtime and a slower, more creative, method that would reduce downtime to a minimum and let us test as we went along. Our original plan was to merge a network's machines onto the ethernet switches by announcing some downtime which our telecom department would use to reconnect each hub in each closet to the newly installed ethernet switches. We would have to repeat this ten times. (one for each subnet!) We found a better way.

The Lay(out) Of The LAN(d)

We use the Lucent SYSTIMAX Structured Cabling System to wire our building. This is an example of the 10base-T ''star'' topology that most medium to large buildings use. Each office is wired to a hub in the nearest closet via copper. The hubs in the closets are then connected via fiber uplinks to the basement. If a network spans multiple closets, they all ''meet'' in the basement at a fiber ethernet hub which connects all the uplinks.

Due to our users' office locations, our networks span about ten of our telecom department's wiring closets. For example, our ''quarto-net'' looks like Figure 4.

[4] `vif`, originally by John Ioannidis, 1991. Modified by many. See http://fy.chalmers.se/~appro/VIF.html

Our ''lexicon-net'' connected a different, but overlapping set of closets. (Imagine 10 major networks and 20-odd small networks overlapping in those closets... and our users represent only 10% of the people in this building.)

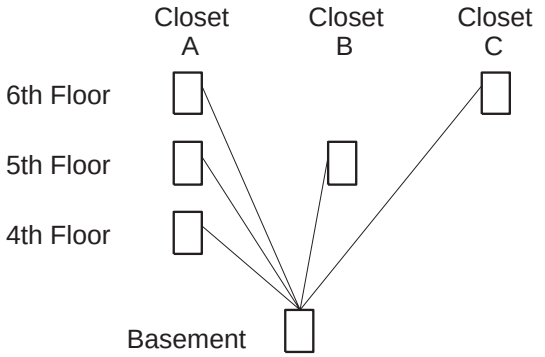

Figure 4: The ''quarto-net.''

A typical closet looks like Figure 5. This diagram represents a closet with many lexicon-net users and a smaller number of quarto-net users right before we migrate users to the ethernet switch (it is unused, except for its connection to the other switches). The lower lexicon-net hubs connect to the top hub, and the top hub connects to the basement. The three lexicon-net hubs are connected in a star rather than a daisy-chain because the diameter of an ethernet is limited by the number of repeaters (hubs count as repeaters). The quarto-net hubs are in a similar arrangement but the star configuration is less obvious since there are only two.

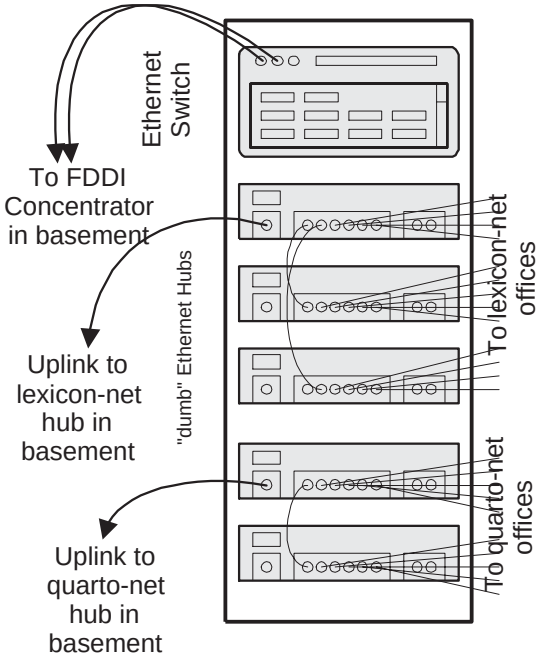

Figure 5: Typical closet.

Migrating To The Ethernet Switch

Here is how we moved users to the ethernet switch with almost no downtime. In the beginning, the switched network (known as ‘‘half-net,’’ because we would eventually split it in half) and lexicon-net are independent networks. They are connected by a router that is in a different part of the building. The router connection for half-net is a dedicated port on an ethernet switch. The router connection for lexicon-net and others is like any other office connection on lexicon-net.

First we connect the entire lexicon-net to half-net at one point (Figure 6). We do that by connecting the top hub in one closet to a port on our switch. This is now two IP subnets on one (switched) ethernet. The router connects at two physically different places (one for half-net, one for lexicon-net) and all data flows as before. Downtime for this maneuver: none.

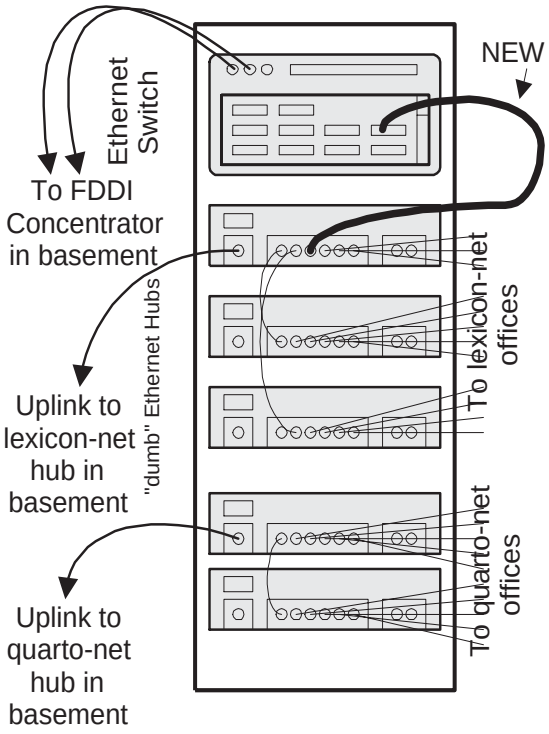

Figure 6: Connection of lexicon-net to half-net.

Now we decommission the router’s lexicon-net interface and assign its old IP address to be the secondary IP address of the router’s half-net interface. Now both half-net and lexicon-net are connected over the same router interface. During this process the machines on lexicon-net will see their router become inactive, then re-appear with a new MAC address. Most TCP/IP implementations cache a MAC address in the ‘‘ARP Cache’’ for 5 minutes and will hang until this value times out and a new ARP is issued to learn about the new router. Downtime can be reduced if the router and clients implement ‘‘Gratuitous ARP.’’ This is where a machine bringing up a new interface broadcasts an ARP packet with the question and answer sections filled in. Most clients will use this information to replace whatever is in their ARP Cache. While clients should implement this, they rarely do. To minimize the distruption, we usually did this at night. It also helped that we manually cleared the ARP caches on the servers and routers right after making the change. Downtime for this maneuver: 5+ minutes

Now that the router has been moved, we are free to move all other hubs onto the ethernet switch at any pace using the following algorithm:

1. Disconnect the hub from the rest of lexicon-net.
2. Connect it to the ethernet switch.

It is important to disconnect the hub before you connect it to the switch, or you may create a loop. Since some hubs (the top hub in our figure) are connected to lexicon-net three times (it is used to connect all the other lexicon-net hubs in that closet) it is best to process that hub last. Downtime for this maneuver: 10 seconds per hub, plus one second for the ethernet switches to ‘‘find’’ the change.

Once the first step of this began, users machines could be renumbered. If this migration took a long time to complete it could be done in parallel with the renumbering of the clients. Parallelism is good. Once the router was reconfigured, our telecom department could re-connect the hubs at will. A lot was gained by not having to move lock-step with our telecom department; they were extremely busy with the other networks in the building.

Now that we could renumber any machine at will. We renumbered each to either the new AT&T subnet or the Bell Labs subnet. We used the techniques described later in this paper.

Splitting Hairs

The renumbering took two months and during that time we we collected information about which jacks had machines targeted to AT&T and which had machines targeted to Lucent. We correlated employees targeted to AT&T to their office number, and office numbers with jacks. Who was going to which company was a moving target, but by enlisting the secretaries the process went exceptionally smoothly.

Also during this time our telecom department installed additional ethernet switches in the closets that would have both AT&T and Lucent networks. These AT&T ethernet switches were connected to their own FDDI concentrator, but the two (AT&T and Lucent) FDDI concentrators were connected to make one large ring.

We delivered our list of which jacks were targeted to which company to our telecom department. They set to work grouping users so that hubs connected exclusively AT&T or Lucent endpoints. When they were done a typical closet looked like Figure 7.

The Grand Finale

When the AT&T router had arrived, we would disable the AT&T net secondary IP address on our router and configure AT&T's new router to use that IP address. The routers were connected by a short wire that we could break when ready. Eventually the AT&T router had its own connection to the Lucent/AT&T backbone and the wire wasn't needed. Around the same time, we separated the AT&T FDDI ring from the Lucent FDDI ring. That was the moment of truth when we would find out which machines were not properly renumbered, and which jacks were not connected to the correct hubs. Much to our surprise, we had exactly four machines lose connectivity when we made that last cut. We considered that a huge success!

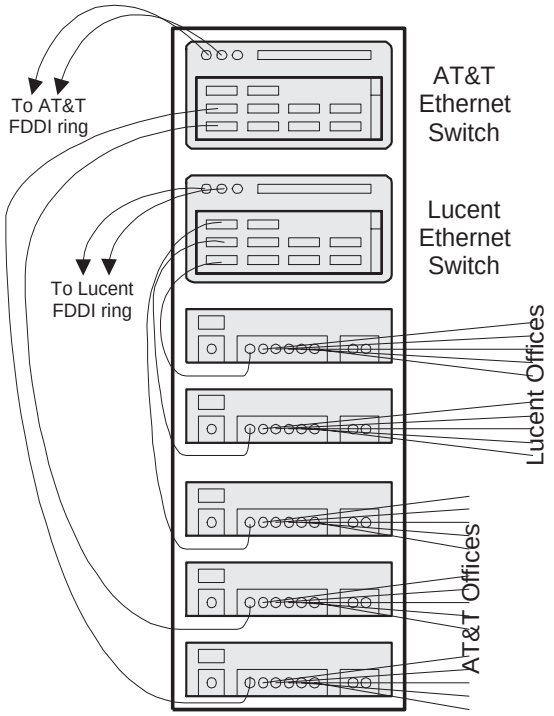

Figure 7: Completed hub schematic.

Then It All Fell Apart

We had some glitches. Some wires were connected incorrectly; sometimes tugging on one wire made another come loose. The subnets of a Class B network must be contiguous, and since the AT&T router for Crawford Hill arrived late, we ended up having to complete the Crawford Hill building first before we could do parts of the Holmdel split. However, none of these problems compared to what happened next.

After the big merge, things really fell apart. Some Sun servers weren't reliable with the `vif` driver, so we used spare ethernet cards to give the machines two physical ports. Sun workstations assign the same MAC (ethernet) address to both interfaces.[5] This confused our ethernet switches and once actually caused a Cabletron ESX-MIM to hang until it was power-cycled. Since it was in a closet owned by our telecom department, we couldn't reboot it until they arrived in the morning.

The big problem was that we underestimated the need of bandwidth to our router. When using secondary IP addresses, packets destined for a different IP subnet (but the same physical ethernet) go up to the router then back down the same router port along the same physical ethernet. While we realized that this would mean an extra load on the router, the load was too heavy and the network fell apart. Machines were nearly unusable if they had to access a fileserver via the router. They all did. We felt that dedicating multiple switched ports to the router would solve the problem but it didn't. (Neither the router nor ethernet switch vendor could explain why, but they both kept explaining that it should work.)

We don't know why we didn't expect this and upgrade the bandwidth to the router, but the fact is that we didn't, and we paid heavily for this. Now FastEthernet is commonplace, but it wasn't then. In hindsight we realize that every packet from every non-renumbered client would be going through the router. That would be 90% of all packets at the beginning of the conversion! Obviously, that is more than a 10M ethernet can handle.

The Trouble Began

Users were extremely upset and got management involved. They demanded we fix the problem before we moved on. However, we felt the best solution was to move on because completing the renumbering would fix the problem and serve the need to move the project forward at the same time. We went back and forth on this and management started hinting that they wanted to see proof that the final configuration would work at all. This would be impossible because we had no way to prove that except with sketchy Sniffer plots and hand waving.

We did not want to take any steps backwards because we felt it was difficult enough to make progress and if we permitted one slide backwards to fix a problem, we would end up sliding back in other ways and eventually the split would not be completed at all; then corporate management would be really upset.

We call this "The Broken Network Conundrum." This is where you have to decide between fixing things and explaining to users why they don't work, because you don't have time for both. One of us

[5]The standards are unclear on how multihomed hosts should behave in this situation. Sun assigns the same MAC address to all ethernet ports of a host. Other vendors assign different MAC addresses per interface.

often muttered, ‘‘I’m too busy mopping the floor to turn the faucet off.’’ This was about the time that many of us were ready to quit.

The result is that we spent about three weeks with a network that was unusable to many of our users. During that time we had meetings with vendors, generated reports with sniffers, chased non-problems, etc. Meanwhile, we renumbered the users that complained heavily so that they were on the same IP subnet as the machines they accessed, therefore eliminating the problem for them. Eventually the complainers managed to all get their machines renumbered, which was our original proposal for fixing things.

Finally we could move on.

Storming The Hallways

We were permitted to continue once we reached a point where the intolerable complainers had been satisfied and the others began to understand that moving forward would fix the problem. At this point they had needlessly spent three weeks with a nearly unusable network. Conclusion: once you renumber the servers, renumber the clients immediately. We were so exhausted after renumbering the servers that we didn’t begin the clients right away. Considering that certain groups of clients communicated mainly with certain servers, we could have done the renumbering one cluster at a time (the server followed by its clients).

We announced a calendar (Listing 3) of when each hallway would be converted. Each week consisted of converting a number of hallways on Monday and Wednesday, giving us a day in between to fix problems that arose. We made no changes on Friday in hope that we might sleep easy on the weekends. We warned users that their hallway’s date would only be changed if they could convince another hallway to swap with them. Most hallways were almost entirely the same department and gladly accepted the calendar as an excuse to plan alternative activities. One department held a picnic.

On the days set aside for changes, we used what we called ‘‘The Rioting Mob Technique.’’ At 9 A.M. we would stand at one end of the hallway. We’d psych ourselves up, and move down the hallways in pairs. At each office we kicked the users out of the office and went machine to machine making the needed changes. Two pairs were PC admins, two pairs were Unix admins. (Each pair either did the left or right side of the hallways). The Unix script was quite robust but sometimes broke, or the tar file was too large for /tmp, or becoming ‘‘root’’ on the machine was difficult. Rather than trying to fix it themselves, the SA would call the senior SA that wrote the script to fix the problem and move on to the next machine. Meanwhile a final pair of SAs stayed at our ‘‘command central’’ where people could phone in requests for IP addresses, provide updates to our host inventory, the host table, etc.

We spent the next day cleaning up anything that had broken. On this ‘‘breather’’ day we also met to refine the process. After a brainstorming session determined what went well and what needed improvement, we determined it was better to make one pass through the hallway calling in requests for IP addresses, giving users a chance to log out and identifying non-standard machines for the senior SAs to focus on. The second pass through the hallway everyone had the IP addresses they needed and things went more smoothly. Soon we could do two hallways in the morning and do our cleanup in the afternoon.

The brainstorming session between the first and second conversion day was critical as everyone had excellent suggestions about how to improve our process. On subsequent breather days we still met but now that our process was refined, our meeting was used to pre-plan for the next day. Many times a conversion day went smoothly enough that we were done by lunch, had the problems resolved by the afternoon, and spent our breather day on other projects.

Other Changes

Meanwhile many other systems needed to be cloned, moved or functions disbanded. This included email, news and many DNS-related issues. These issues could fill another paper. We will not document them because they are very specific to our site and most of what we did was non-inventive.

Communication Is Key

We held weekly ‘‘user feedback sessions’’ to answer questions and give status and ‘‘heads up’’ information. This made users feel included in the process, which increased their cooperation. They also provided excellent feedback about what they felt was important.

Reward Those Who Helped

Eventually, the IP portion of the split was complete. We still had to split the passwd files, duplicate some license servers, and clean up a million small issues before the corporate routers would no longer pass packets between the two companies. But the big physical network split was finally over. We felt it was important to reward those that helped with the project. Our management surprised the technicians from the telecom department by awarding them bonuses. Even though we were only 1/10th of their users in this building and the other 90% were less technically sophisticated and required them to do a lot more of the ground work, we believe we were the only ones to reward them so. Later both AT&T Labs and Lucent Bell Labs both rewarded us similarly.

Related Topics

The PC Movement

DHCP saved us. When we stormed the hallways the PC SAs usually simply clicked ‘‘Use DHCP’’ and rebooted. DHCP is a protocol extension to BOOTP which permits machines to get just about every network parameter they might need from the network. This lets us maintain a central database of parameters which can be updated globally. PCs receive their updated parameters the next time they reboot. We wished Unix could use DHCP so well.

NCD X Terminals

Our NCDs had four ‘‘classes’’ of configurations (one of which being ‘‘every man for himself’’). We developed one solid and maintainable configuration that worked everywhere and changed all NCDs to use it. We now store no configuration on the terminal and use TFTP for all the data. Each client’s configuration file is simply a symbolic link to the master configuration file. We can make a global change by modifying the master file and waiting for all the NCDs to be rebooted. A simple script was written to create the right symbolic links so that the error-prone task of converting IP addresses to hex, and typing the extremely long path names was eliminated.

What We Would Do Differently

In hindsight, there are some things we could have done differently. We could have renumbered just the AT&T machines, not all of them. We (Lucent) were the majority. However, this option would not have let us use this opportunity to renumber to our new Class B IP network, do our much-needed Unix re-configuration, and upgrade our network performance, etc. However, it would have reduced our work to almost nothing more than changing our DNS and NIS domains. However, it was not an obvious option to us because AT&T was short staffed and we felt obligated to back-fill for them. We would have had to do much of the work anyway.

We also could have simply declared ownership of the network and gave AT&T a cut-off date for when they had to have all their machines removed. Bell Labs Murray Hill used this technique . . . the lucky dogs! Again, AT&T’s staff shortage would have made this a problem.

“At least it won’t happen again”

We felt oddly disappointed that we had learned so much and developed so many techniques but none of them would be useful in the future. However, as luck would have it, since the split we have found that soon we will need to renumber three more user communities, totaling approximately the same number of machines involved in the split. Imagine our joy and surprise.

What We Learned

- Massive renumbering projects are increasing in frequency due to cleansing of organic networks and corporate structural changes.
- Massive renumbering projects are an excellent opportunity to clean up a messy network.
- It is possible to renumber a massive number of machines without a single day (or time period) where all machines are down.
- After the first day of mass conversion, meet to review areas of improvement and update the process.
- Secondary IP addresses are a useful transition aid, but don’t over-do it.
- Clients have dependencies on servers, so be creative about renumbering servers, and do them early (or add secondary IP addresses).
- Develop ‘‘the perfect machine’’ before you make those changes everywhere else. Make all changes at once to a machine before you move on, don’t make one change to all machines before you move to the next change. This prevents the situation where you convert all machines at once to discover that every single machine now has the same flaw.
- Communicate with your users any way and every way you can.
- The physical split of the network can be easier if you have a solid, structured, wire plant; you know the structure is good when it documents itself.
- Avoid ‘‘the conundrum’’; trust your gut.
- Automated processes should be as simple as needed, but no simpler. Centralize things that should be centralized, but no ‘centraler’.
- Yell a loud chant before you storm the hallways. It psyches you up and makes your users more willing to get out of the way.

Conclusion

On the network side, we used secondary IP addresses to ease the renumbering of the networks. On the management side, we unified our management files so updates are now easy and error-proof (the $M directory and ‘‘`ypmake`’’ script). On the software (operating system) side, we came up with a good automated method of implementing this split, with a larger purpose in mind: our ‘‘reconfig’’ script is still used to change the configuration of SunOS and other systems that don’t support Jumpstart. The ‘‘setupncd’’ script lives on. On the human side, we used communication (weekly meetings/forums, web pages, email broadcasts), and terror (‘‘the rioting mob technique’’).

‘‘Normal’’ work for us nearly ended for nine months as we split the network. We did not take one mess (which every large network grows into) and make it two separate messes. Instead we actually took a tough task (splitting the network) and in the process

of accomplishing it, made things better. These were not two simultaneous but unrelated events. Instead, we came up with a means to do them simultaneously in such a way that they complement and ease the difficulty of each other. The new network is significantly less labor-intensive to manage because we were able to put in place unified policies and procedures.

Acknowledgements

Huge projects like this do not get completed without the help and dedication of the entire team or in this case two teams: all the system administrators involved both from AT&T and Lucent. Our telecom department performed great feats of heroism, especially Dave Wilson. Our significant others and families deserve credit for enduring our long hours and frazzled nerves. We'd also like to thank the users that suffered through it all. Special thanks to Sherry McBride and Mike Richichi for their advice on how to structure this paper.

Availability

Some of the tools created for this project are available on http://www.bell-labs.com/user/tal

References

[Albitz1996] *DNS and BIND*, By Paul Albitz and Cricket Liu, 2nd Edition December 1996, O'Reilly and Associates.

[Hess1992] David Hess, David Safford, Udo Pooch, ''A Unix Network Protocol Security Study: Network Information Service'', *ACM Computer Communications Review* 22 (5), 1992. ftp://net.tamu.edu/pub/security/TAMU.

[Lear1996] ''Renumbering: Threat or Menace?'' Eliot Lear, Jennifer Katinsky, Jeff Coffin, & Diane Tharp, Silicon Graphics, Inc., published in the *USENIX Association's Proceedings of the Tenth Systems Administration Conference (LISA X)*, pp. 91-96.

[Remy1994] ''Tenwen: The Re-engineering of a Computing Environment'', Remy Evard, *Proceedings of the Eighth USENIX Systems Administration Conference Proceedings*, pages 37-46, 1994.

[RFC1900] *RFC1900, Renumbering Needs Work*, B. Carpenter & Y. Rekhter. February 1996.

[RFC1916] *RFC1916: Enterprise Renumbering: Experience and Information Solicitation*, H. Berkowitz, P. Ferguson, W. Leland, & P. Nesser. February 1996.

[RFC2071] *RFC207: Network Renumbering Overview: Why would I want it and what is it.* P. Ferguson, H. Berkowitz. January 1997.

[Stern1991] *Managing NFS and NIS*, Hal Stern, June 1991, O'Reilly and Associates.

[Wietse1996] ''The Replacement Portmapper'', Wietse Venema, <ftp://ftp.win.tue.nl/pub/security/portmap_*>, 1996.

Author Information

Tom Limoncelli is a MTS at Bell Labs, the R&D unit of Lucent Technologies, where he is chiefly concerned with the architecture and operation of the data network for much of Research. Tom started doing system administration on VAX/VMS systems in 1987 and switched to Unix in 1991. He holds a B. A. in C. S. from Drew University, Madison, New Jersey. His homepage is http://www.bell-labs.com/user/tal; he can be reached at <tal@lucent.com>.

Tom Reingold studied music (voice, performance) at Boston University from 1978 through 1980 and computer science at Hunter College of the City University of New York from 1982 through 1987. He has worked both as a programmer and as a system administrator in a variety of environments, ranging from business support, commercial software development, and research. He currently works as a system administrator for the Software and Systems Research Center at Bell Labs. His homepage is http://www.bell-labs.com/user/tommy; he can be reached at <tommy@bell-labs.com>.

Ravi Narayan is a contractor at Lucent Technologies Bell Labs, working on systems and networks administration. He is finishing up his Master's in Computer Science at Worcester Polytechnic Institute, and holds a Bachelor's in Mathematics. His interests include Distributed Systems and Networking, automation of administration, hacking in Perl and the World Wide Web (authoring and administration, since 1993, and mentioned in technical magazines such as PC Computing: 1001 Best Web Sites, Dec 94). His homepage is http://www.bell-labs.com/user/rn; he can be reached at <rn@bell-labs.com>.

Ralph Loura is a Technical Manager at Bell Labs Research, the Research arm of Lucent Technologies, where he is responsible for managing the team providing computing and networking services to about half of Bell Labs Research. Ralph has been working with UNIX Systems since the mid eighties in jobs that have included O. S. programing, device driver development, distributed applications programming, systems administration, systems engineering, computing environment architecture and strategy, vendor management and financial planning. He holds a B. S. in C. S. and Mathematics from Saint Joseph's College, Rensselaer, IN and a M. S. in C. S. from Northwestern University, Evanston, IL. His homepage is http://www.bell-labs.com/user/ralph; he can be reached at <ralph@lucent.com>.

Listing 1: The NIS Master's Makefile

```
# The NIS Makefile
# Master copy: milk:/var/yp/Makefile
# Source Control: RCS
# Distribution: none (it stays here)

[much deleted]

B=-b
#B=
#VERBOSE=
VERBOSE=-v
#DIR =/etc
DIR =/var/yp/master
DOM = `domainname`
NOPUSH = ""
#NOPUSH = 1
#ALIASES = /etc/aliases
ALIASES = $(DIR)/aliases
YPDIR=/usr/etc/yp
YPDBDIR=/var/yp
YPPUSH=$(YPDIR)/yppush $(VERBOSE)
MAKEDBM=$(YPDIR)/makedbm
REVNETGROUP=$(YPDIR)/revnetgroup
STDETHERS=$(YPDIR)/stdethers
STDHOSTS=$(YPDIR)/stdhosts
MKNETID=$(YPDIR)/mknetid
MKALIAS=$(YPDIR)/mkalias

# Local defines
DNSCONVERT=/var/named/bin/h2n
MKNETGROUP=/usr/local/adm/bin/mk.netgroup
NETGROUPCONF=$(DIR)/netgroup.config

CHKPIPE=  || (  echo "NIS make terminated:" $@ 1>&2; kill -TERM 0 )

k:
    @if [ ! $(NOPUSH) ]; then $(MAKE)  $(MFLAGS) -k all; \
    else $(MAKE) $(MFLAGS) -k all NOPUSH=$(NOPUSH);fi

all: passwd group ethers hosts web.done networks rpc services protocols \
    netgroup.time bootparams aliases netmasks \
    auto.master auto_master auto.home auto_home timezone \
    ypservers.time hosts.mail.time dns.time toaster.time netid \
    legacy.time attlabs.time

[ the following are not deleted as they are unchanged from
the sample that comes with SunOS 4.1.4: passwd.time, group.time,
hosts.time, ethers.time, networks.time, services.time,
rpc.time, protocols.time, netgroup.time, bootparams.time,
aliases.time, netmasks.time, netid.time, timezone.time ]

auto.master.time:  $(DIR)/auto.master
    -@if [ -f $(DIR)/auto.master ]; then \
        sed -e "/^#/d" -e s/#.*$$// $(DIR)/auto.master \
        | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/auto.master; \
        touch auto.master.time; \
        echo "updated auto.master"; \
        if [ ! $(NOPUSH) ]; then \
            $(YPPUSH) auto.master; \
            echo "pushed auto.master"; \
        else \
        : ; \
```

```
        fi \
    else \
        echo "couldn't find $(DIR)/auto.master"; \
    fi

[ the following maps are not included because they are extremely
similar to auto.master.time: auto_master.time, auto.home.time,
auto_home.time ]

# The NIS servers are listed in $M/ypservers
# (Now we can edit $M/ypservers rather than using "ypinit -m"
# to edit our list of slaves)
ypservers.time:  $(DIR)/ypservers
    -@if [ -f $(DIR)/ypservers ]; then \
        sed -e "/^#/d" -e s/#.*$$// $(DIR)/ypservers \
        | egrep -v "^\s*$$" \
        | (awk '{ print $$1 " " $$1 }') \
        | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/ypservers; \
        touch ypservers.time; \
        echo "updated ypservers"; \
        if [ ! $(NOPUSH) ]; then \
            $(YPPUSH) ypservers; \
            echo "pushed ypservers"; \
        else \
        : ; \
        fi \
    else \
        echo "couldn't find $(DIR)/ypservers"; \
    fi

# This generates DNS data from the NIS hosts file.

dns: dns.time
dns.time: $(DIR)/hosts $(DIR)/dns.opts
    (cd /var/named && make)
    touch dns.time

# This generates netgroup data from the $M/netgroup.config file

$(DIR)/netgroup: $(DIR)/hosts $(DIR)/netgroup.config $(MKNETGROUP)
    $(MKNETGROUP) -d sun1134      ${NETGROUPCONF} >$(DIR)/netgroup.sun1134
    $(MKNETGROUP) -d sun1135      ${NETGROUPCONF} >$(DIR)/netgroup.sun1135
    $(MKNETGROUP) -d hoh          ${NETGROUPCONF} >$(DIR)/netgroup.hoh
    $(MKNETGROUP) -d ''           ${NETGROUPCONF} >$(DIR)/netgroup.toaster
    $(MKNETGROUP) -d info.att.com ${NETGROUPCONF} >$(DIR)/netgroup

toaster.time: toaster.host.time toaster.netgroup.time
    touch toaster.time

# Push hosts to the NAS FAServers
toaster.host.time: $(DIR)/hosts
    mntdir=/tmp/nac_etc_mnt_$$$$;\
    if [ ! -d $$mntdir ]; then rm -f $$mntdir; mkdir $$mntdir; fi;\
    for faserver in `cat $(DIR)/toasters` ; do \
        echo Pushing hosts to $$faserver;\
        mount $$faserver:/etc $$mntdir;\
        cp $$mntdir/hosts $$mntdir/hosts.bak;\
        cp $(DIR)/hosts $$mntdir/hosts.new;\
        mv $$mntdir/hosts.new $$mntdir/hosts;\
        umount $$mntdir;\
    done;\
    rmdir $$mntdir
    touch toaster.host.time
```

```
# Push netgroup to the NAS FAServers
toaster.netgroup.time: $(DIR)/netgroup
    mntdir=/tmp/nac_etc_mnt_$$$$;\
    if [ ! -d $$mntdir ]; then rm -f $$mntdir; mkdir $$mntdir; fi;\
    for faserver in `cat $(DIR)/toasters` ; do \
        echo Pushing netgroup to $$faserver;\
        mount $$faserver:/etc $$mntdir;\
        cp $$mntdir/netgroup $$mntdir/netgroup.bak;\
        cp $(DIR)/netgroup.toaster $$mntdir/netgroup.new;\
        mv $$mntdir/netgroup.new $$mntdir/netgroup;\
        umount $$mntdir;\
        rsh -n $$faserver exportfs -av;\
    done;\
    rmdir $$mntdir
    touch toaster.netgroup.time

# Push the hosts.mail file

hosts.mail.time: $(DIR)/hosts.mail
    rcp $(DIR)/hosts.mail octavo:/etc/hosts.mail
    touch hosts.mail.time

passwd: passwd.time
group: group.time
hosts: hosts.time
ethers: ethers.time
networks: networks.time
rpc: rpc.time
services: services.time
protocols: protocols.time
bootparams: bootparams.time
aliases: aliases.time
netid: netid.time
netmasks: netmasks.time
timezone: timezone.time
auto.master: auto.master.time
auto_master: auto_master.time
auto.home: auto.home.time
auto_home: auto_home.time
$(DIR)/netid:
$(DIR)/timezone:
$(DIR)/auto.master:

# Push things to the legacy systems
#    (this is at the end of the file to make it easy to delete...
#      this will be deleted someday, right?)

legacy.time: legacy.hosts.time legacy.aliases.time \
    legacy.auto_master.time legacy.auto.home.time legacy.auto_home.time \
    legacy.netgroup.peerless.time legacy.netgroup.boole.time \
    legacy.netgroup.octavo.time
    rsh -n octavo "cd /etc && /bin/make"
    touch legacy.time

legacy.hosts.time: $(DIR)/hosts
    rcp $(DIR)/hosts octavo:/etc/hosts.common
    touch legacy.hosts.time

legacy.aliases.time: $(ALIASES)
    rcp $(ALIASES) octavo:/etc/aliases.common
    rcp $(ALIASES) boole:/etc/aliases
    rcp $(ALIASES) learnx:/etc/aliases.common
    rcp $(ALIASES) caribbean:/etc/mail/aliases.common
```

```
    rsh caribbean "cd /etc/mail && /usr/ccs/bin/make install"
    touch legacy.aliases.time

legacy.auto_master.time: $(DIR)/auto_master
    rcp $(DIR)/auto_master octavo:/etc/auto_master
    touch legacy.auto_master.time

legacy.auto.home.time: $(DIR)/auto.home
    rcp $(DIR)/auto.home octavo:/etc/auto.home
    touch legacy.auto.home.time

legacy.auto_home.time: $(DIR)/auto_home
    rcp $(DIR)/auto_home octavo:/etc/auto_home
    touch legacy.auto_home.time

legacy.netgroup.peerless.time: $(DIR)/netgroup.hoh
    rcp $(DIR)/netgroup.hoh peerless:/etc/netgroup
    touch legacy.netgroup.peerless.time

legacy.netgroup.boole.time: $(DIR)/netgroup.sun1134
    rcp $(DIR)/netgroup.sun1134 boole:/etc/netgroup
    touch legacy.netgroup.boole.time

legacy.netgroup.octavo.time: $(DIR)/netgroup.sun1135
    rcp $(DIR)/netgroup.sun1135 octavo:/etc/netgroup
    touch legacy.netgroup.octavo.time

# Push things to the attlabs systems then do their "ypmake"
#    (this is at the end of the file to make it easy to delete...
#      this will be deleted someday, right?)

attlabs.time: attlabs.hosts.time attlabs.ethers.time \
    attlabs.auto_home.time attlabs.netgroup.config.time
    rsh -n shy "cd /var/yp && /usr/local/bin/flock \
        /var/tmp/ypmake.lock /usr/bin/make"
    touch attlabs.time

attlabs.hosts.time: $(DIR)/hosts
    rcp $(DIR)/hosts shy:/var/yp/master/hosts
    touch attlabs.hosts.time

attlabs.ethers.time: $(DIR)/ethers
    rcp $(DIR)/ethers shy:/var/yp/master/ethers
    touch attlabs.ethers.time

attlabs.auto_home.time: $(DIR)/auto_home
    rcp $(DIR)/auto_home shy:/var/yp/master/auto_home
    touch attlabs.auto_home.time

attlabs.netgroup.config.time: $(DIR)/netgroup.config
    rcp $(DIR)/netgroup.config shy:/var/yp/master/netgroup.config
    touch attlabs.netgroup.config.time

# This generates the web page that lists which machines
# are targeted to which company

web:web.done
web.done: /home/adm/bin/mk.tco-webpages $(DIR)/hosts $(DIR)/tco.machines
    /home/adm/bin/mk.tco-webpages | \
        rsh xxx "cat >/home/tal/public_html/tco.html"
    touch web.done
```

Listing 2: mk.netgroup

```
#!/opt/perl5/bin/perl

# mk.netgroup -- convert the netgroup.config file to NIS's netgroup format.
#  by tal

# Note: a lot of this comes from THE PERL DATA STRUCTURES COOKBOOK
# http://www.perl.com/perl/pdsc/index.html (esp. "Hashes of Lists" examples)

$MAX_LINE = 252;    # max linelength of a NIS data record

# Process options

$opt_d = '';    # NIS domain.
$opt_m = '';    # special NIS-less rpc.mountd (like on volume)
require "getopts.pl";
do Getopts('d:m');

# read from file data structured like:
# flintstones: fred barney wilma dino
while (<>) {
    # the order of the next three lines is very important
    s/#.*//; # toss comments
    next if /^\s*$/; # skip blank lines
    redo if s/\\$// and $_ .= <>; # handle continuations

    next unless s/^(.*?):\s*//; # parse (and skip badly-formatted lines)
    $host = $1;
    die "Host/group $host is listed twice.  Aborting." if $seen{ $host }++;
    foreach $item ( split ) {
        # is this a group or a host
        if ($item =~ /^\+/) {
            $item =~ s/^\+//g;
            push @{ $netgroup{$host} }, $item;
            # add $item to %GROUPS
            $GROUPS{ $host } = 1;
            $GROUPS{ $item } = 1;
        } else {
            # add the host to that list
            if ($opt_m) {
                push @{ $netgroup{$item} }, "$host";
            } else {
                push @{ $netgroup{$item} }, "($host,,$opt_d)";
            }
            # record that such a group exists
            $GROUPS{ $item } = 1;
        }
    }
}

# write out the netgroup file
foreach $group ( sort keys %netgroup ) {
    &one_group( 0, $group, sort @{ $netgroup{$group} } );
}

exit 0;

# This takes a netgroup name and a list of what should appear
# on its line and prints it.  If the line is too big, it splits
# it into "group_1", "group_2", etc. and generates
# "group: group_1 group_2", etc.
sub one_group {
    local($count, $g, @l) = @_;
    local($line);
```

```
    $origcount = $count;
    local(@m) = ();
    $max = $MAX_LINE - length($g) - 5;
    $line = shift @l;
    foreach $i ( @l ) {
        if ((length($line) + length( $i )) > $max) {
            $count++;
            print $g, "_", $count, "\t", $line, "\n";
            push @m, "${g}_$count";
            $line = $i;
        } else {
            $line .= " " . $i;
        }
    }
    if ($count ne $origcount) {
        $count++;
        print $g, "_", $count, "\t", $line, "\n";
        push @m, "${g}_$count";
        &one_group( $count, $g, @m);    # LOOK!!! RECURSION!!!
    } else {
        print $g, "\t", $line, "\n";
    }
}
```

Listing 3: The Announcement

```
To: (all users)
From: help@big.att.com
Subject: Holmdel/Crawford Hill network split is coming! (READ THIS!!!)

Summary:  We will be stopping by to reboot your machine based on the
schedule below.  No exceptions.  It is required for the Lucent/AT&T
Split.  After your machine has been touched, email from it will either
be "From: user@research.att.com" or "From: user@dnrc.bell-labs.com"
depending on what company the machine is targetted to.  All other
changes should be transparent.

IF YOU WILL NOT BE IN ON YOUR SCHEDULED DAY, PLEASE LOG OUT
THE NIGHT BEFORE.  DO NOT LEAVE YOUR SCREEN LOCKED.

We will be attacking machines on a hallway-by-hallway basis according
to this schedule:

Mon 6/10   5th floor B-E Aisles
Tue 6/11   "fishnet" (5th floor G Aisle)
Wed 6/12   3th floor G Aisle ("neural net")
Mon 6/17   6th floor A-G Aisle
Mon 6/18   "signal-net" users (4th floor)
Wed 6/19   5th floor F-G Aisles ("boole net")
Mon 6/24   All of HOH
6/5-6/15   Spin net users in dept1136 will be spread over these days.

The details:

Before we can split the "big" network (also called the "info.att.com"
network), we must prepare each machine.  This preparation is mandatory
and must be completed by June 26.  It requires that we reboot your
machine.  It will go fastest if we do one hallway at a time.

When we are done you should have as much functionality as you did
before.  The one thing you will notice is that email sent from that
machine will have a From: header that lists
either "From: user@research.att.com" or "From: user@dnrc.bell-labs.com"
```

depending on what company the machine is targetted to. (you will receive email with more details)

If you find something no longer works we need you to report it to "help@big.att.com" as soon as possible or stop us while we are in your hallway converting other machines. You can also call:
HO 908-949-xxxx, 908-949-xxxx, 908-949-xxxx
HOH 908-888-xxxx

We expect some network instability during this process. Be forewarned.

There is a chance that your password may revert back to an older password, if this happens contact us and we can fix this quickly.

NOTE TO LUCENT EMPLOYEES: Lucent users can log into "shelf" (a Solaris 2.5 Sparc 20) or "hyper" (a SunOS 4.1.4 Sparc 1+) to see what a converted machine is like. You shouldn't feel any differences. Please report any problems you find, etc.

NOTE TO AT&T EMPLOYEES: We will soon announce machines that you can log into to see that the new environment works for you.

NOTE TO 1136 "SPIN" EMPLOYEES: You will be contacted individually about your conversion. Access to "shelf" and "hyper" (mentioned above) for you will be ready soon.

VOLUNTEER REQUEST: We would like to convert some machines a couple days early so we'll discover problems specific to your department/area/group. If one courageous Sun user and one dare-devil PC user from each hallway would like to be an "early adopter" it will save the rest of your department a lot of pain. Send email to "help" to volunteer.

ARE WE JUST REBOOTING? CAN I DO THAT MYSELF? No, we're doing a lot more than rebooting your machine. We're installing new sendmail configurations, DNS, automount, and a whole heck of a lot more.