Exploiting Secondary IP Addresses for Fun And Profit: Creating the network for Lucent Bell Labs Research New Jersey South

(Note to LISA Committee: We need help picking a better title)

EXTENDED ABSTRACT

Thomas Limoncelli, Thomas Reingold, Ravi Narayan, Ralph Loura
Lucent Bell Labs
Holmdel, NJ, USA

Abstract

This paper describes the tools and techniques used to split the AT&T Bell Labs Research networks in Holmdel and Crawford Hill facilities into separate networks for Lucent Bell Labs and AT&T Labs as part of the “trivestiture” 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 also 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. This paper is from the perspective of the Lucent Bell Labs system administrators, not the AT&T Labs SAs. The transition did not go smoothly, and if we could have read this paper before we began we could have avoided many of the problems.

The Increasing Trend

There is little literature on the topic of merging, splitting, and renumbering computer networks because it is infrequently done and 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:

- IP Address Shortages have caused many sites to renumber.
- Old networks need to be cleaned. We see a growing trend in research and academic environments where networks had grown organically and reach a point where they must be straightened out.
- Networks that grow this way often do so because they are renegades or placed outside of any central control. In recent years corporate CIO departments have downsized from mainframes to Unix environments. Now the suits use what was once the domain of the renegades and soon centralization or at least standardization becomes an obvious way to save money, or is done for political reasons. This often results into merging networks.
- The Reagan Era was marked by constant corporate buy-outs. Now some monolithic companies are splitting themselves. We feel that corporate divestitures such as AT&T's may be a growing trend. Watch out Microsoft!

1. Or at least an insignificant minority.

The old way to split networks:

From informal surveys, we found that most sites perform mergers and splits in a 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. This 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 preferred the first option, but management wouldn’t approve a simultaneous vacation of approximately 500 researchers. The second option was too labor intensive. We adopted a hybrid technique.

The Task:

[Here the paper explains the split of AT&T into AT&T, Lucent, and NCR. The scope of our paper does not include corporate computing services. We also explain why our network was such a mess (its all history).

[We had already planned a major network re-organization, now we took that plan and used it to plan the split: i.e. it would be easy to do the split as we did the reorg)

[The corporate CIO group took care of the backbone routers and central services. We didn’t have to.]

What our network looked like:

The network consisted of four main user communities each with its own SA procedures and standards. They had never been properly merged, but we had eliminated

90% of the UID conflicts over time. There were four NIS domains, 2 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 networks and a newly allocated Class B which we divided into subnets and were migrating to slowly. Almost all of our machines were connected via 10base-T wiring which is administrated by our telecom department (Cat 3 copper is Cat 3 copper, right?) with 2-3 pockets of thinnet. Before the split was announced, we wanted to build 2 major networks for Bell Labs, one in each building, with many small subnets for experiments only. We wanted no more private networks to boost egos. The major networks would consist of etherswitches tied together with FDDI or other high-speed network topology. We wanted one NIS domain, one /usr/local (or equivalent), one procedure for everything, etc. 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, etc. cut the networks, and be done.

What we did Part 1: NIS Domains and Stranger Things

Changing the IP address of a machine affects all the machines that depend on it for services (i.e. 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 3 of the NIS domains into an existing fourth, we created a new NIS domain and merged all four into this new domain. It was more work this way, but permitted us to start with a clean NIS configuration that we could experiment with, install security-enhancement tools, and develop modern update procedures, etc. It also let us start with a fast machine, since future upgrades would be difficult.

The new NIS Configuration

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 call $M) and use xed to edit the files.

[xed is a wrapper around $EDITOR that locks the file and uses RCS. “ypmake” is a script that runs the NIS makefile but only lets one person at a time run it]

The benefits of these changes 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”.
- The makefile encapsulates all the update procedures in one place. Easier to debug.
- Better security on the NIS master: While $M/passwd contained the real passwd file, and /etc/passwd contained a minimal password file. Break-ins that steal the /etc/password file would be useless.

[We got creative about putting many things in $M and using the Makefile to drive all of our update processes.]

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.

NIS slaves were configured to serve the NIS databases of the old and new master at the same time.[1] Once this was done, individual clients could be converted to the new NIS domain for testing before “the big cut-over”. [We found it important to “keep things simple” when writing our scripts. The legacy scripts were un-fixable because they were too complicated.]

What we did Part 2: Building the perfect pair(s)

With the servers successfully tested we decided to put them into use for the features that were ready. This was a big mistake. We spent a number of days trying to change every machine to the new DNS servers. If we had to do that for every change, we would never be

1. Refer to Hal Stern’s NIS book here

done. Worst of all, if the change had been incorrect, we would have had to do it all over again.

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.

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.

[Here we should have an example of using the script. Show that it takes two arguments: the new IP addr and hostname; from there is can 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]

(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)

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.

What we did Part 3: 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 the NEU tenwin paper, we had money to burn, not little 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 the "ypmake" makefile converted into a real netgroup file. The format looks like this:

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

This would generate a netgroup 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 places, all the machines were classified with a single group, and partitions were exported to either "bl" or "att" but not "allnfs". These tasks could be divided over many SAs and done in parallel. Changing the IP address of a fileserver means rebooting all of its NFS clients. We avoided this by using vif[1], a device driver that lets a SunOS machine appear on two IP addresses at once. As clients rebooted due to unrelated reasons, we would eventually have all machines talking to the new IP address. Solaris servers can do this using secondary IP addresses (le0 becomes le0:0 and le0:1) each with their own IP address.

What we did Part 4: Merging the IP ranges, splitting the wires

[This is where I will explain how secondary IP addresses work, and why it is possible to have two (or

1. vif is by JI, get real footnote info

more) IP subnets running on the same wire just like EtherTalk, DECnet and IP can share an ethernet.]

The physical split was designed to be neat and orderly after a chaotic merge: To split the networks we planned on merging the main production networks into one major network in each building [this would work because we used secondary IP addresses. We could then renumber “at will”, then move machines to the right ethernet hubs in the closets, then move the hubs to the right etherswitches, then split the etherswitches between two router ports, then move the AT&T router port to their own router, then have the routers talk to each other only through the corporate backbone]

[Some logistics were difficult because a Class B has to be contiguous, but we made due.]

Then it all fell apart:

After the big merge, things really fell apart. Some Suns weren't reliable with the vif driver [and our workarounds were sometimes worse than the original problem]

The big problem was that we underestimated the need of bandwidth to our router. [the paper has details] Machines were nearly unusable if they had to access a fileserver via the router. We felt that dedicating multiple switched ports to the router would solve the problem but it didn't.

We don't know why we didn't expect this and upgraded 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.

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 splitup project forward at the same time. [the paper has more details]

The result is that we spent about 3 weeks with a network that was unusable to many of our users. [the paper has more details. We got though it with microfixes.]

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. As one of us repeated, “I'm too busy wiping the floor to turn the faucet off.” This was about the time that many of us were ready to quit.

What we did Part 5: Storming the hallways

We announced a calendar 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 so we could sleep well 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. Some hallways were almost entirely the same department and gladly accepted the calendar as an excuse to plan alternative activities.

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 to hand out, update what hosts had changed to which IP, etc. We spent the next day cleaning up anything that had broken.

On the ne “breather” day we met to refine the process. After a brainstorming session determined was went well and was needed improvement, we determined it was better to make one pass through the hallway calling in requests for IP addresses and giving users a chance to log out and identifying difficult 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.

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.

What we did Part 6: Communication is key

We held weekly "user feedback sessions" to answer questions and give status and "heads up" information. This made users felt included in the process which made them more cooperative. They also provided excellent feedback about what they felt was important.

Reward Those Who Helped:

[The paper explains how we thanked those that helped us by giving them cash bonuses]

Other topics:

[here the paper explains that we used DHCP extensively with PCs, and made NCD x-terminals use bootp/tftp; all of those machines now can be managed much easier]

What we would do differently:

Renumbered half the machines, not all of them. In hindsight, it would have been much easier, since we (Lucent) were the majority, if we 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! While this was not an option for us, it would not have let us use this opportunity to do the IP renumbering, Unix re-configuration, and network performance enhancements we wanted to do. However, it would have reduced our work to almost nothing more than changing our DNS domain. However, in hindsight we could have renumbered only 40% of the machines -- the AT&T machines -- and left the IP addresses of the Lucent machines alone. In other words, we could have kept status quo on some machines rather than change every single machine. [This paragraph is awkward]

"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 300 machines. Oh goody.

Conclusions:

- Massive renumbering projects are increasing in frequency due to cleansing of organic networks and corporate structural changes.
- 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 change the process.
- Secondary IP addresses are a useful transition aide, 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 has the same flaw.
- Communicate with your users any 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.
- 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.
- Automated processes should be as simple as needed, but no simpler. Centralize things that should be centralized, but no centraler.

References:

"Tenwin: The Re-engineering of a Computing Environment", Remy Evard, Proceedings of the Eighth USENIX Systems Administration Conference Proceedings, pages 37-46, 1994.

RFC1900, Renumbering Needs Work

RFC1916, Enterprise Renumbering: Experience and Information Solicitation

RFC2071, Network Renumbering Overview: Why would I want it and what is it anyway?

(we need to add 3-4 more references here)

O'Reilly DNS Book

O'Reilly Sendmail book

Weiste's "making NIS more secure" paper