Memorandum

Date:	December 13, 1994
From:	Karl Auerbach
To:	LAN Magazine
Subject:	Article -- Tips for diagnosing and repairing TCP/IP networks
Pages:	8

Tips for Diagnosing and Repairing TCP/IP networks.

Introduction

Almost everyone, is rushing to get onto "the Internet". Many people are experiencing TCP/IP for the first time.

TCP/IP networks can be rather complicated and can demonstrate some "interesting" failure syndromes. The purpose of this article is to give you, the reader, a rough framework to use when dealing with TCP/IP problems.

The Strategy

The most effective approach is the familiar "scientific method" combined with a "divide and conquer" strategy. Use your knowledge of the symptoms and your knowledge of how networks operate to formulate hypotheses. Perform tests to confirm which hypotheses may be valid and which are clearly incorrect. Start by making broad hypothesis and then, with each round of the scientific method, begin to slowly "zero in." For example, I would begin with a broad hypothesis, such as "the problem is network connectivity rather than the user's computer." Then I'd perform tests to determine whether the network does, in fact, exhibit a connectivity problem. If so, I might then refine the hypothesis to say something like "the problem is a routing failure".

Success is not guaranteed. However, by using this methodology your chance of success is vastly higher than an unfocused hit-and-miss strategy.

Networks are extremely complex distributed systems with much non-deterministic behavior. It is very easy to make a mistake or miss something when trying to isolate a problem. So if your initial attempt to solve a given problem leads nowhere, take a break and start anew but be more careful about your hypothesis definition and the tests you perform. Be careful not to read too much into any single test and be careful not to prematurely discard a hypothesis. And if things don't seem to be getting anywhere, be willing to re-evaluate your previous conclusions.

The main difficulty of this approach is coming up with a reasonable hypothesis -- the actual testing is easy by comparison. A secondary difficulty is analyzing your tests to determine whether you can really reject or accept a hypothesis.

Think: "What could be causing the symptoms that I am seeing?" This, of course, means that you know what are symptoms and what are simply irrelevant observations. It also means that you have a good feel for how the various elements of a network interact with one another, especially how those interactions go awry when things are not quite in perfect order.

Experience will help. And so will the framework which follows. This framework has been acquired through long (and painful) experience dealing with both healthy and sick networks.

Begin by formulating a hypothesis which tries to determine the general nature or location of the problem. For example begin by trying to determine whether a problem is associated with a given computer or with the network as whole.

Troubleshooting Aphorisms

Most hypothesis-test cycles are mental and don't even involve the network or physical testing at all.

Keep good notes. It can be very frustrating to think you have determined some fact when you have not.

Readily repeatable problems, or total outages are the easiest problems to resolve.

Intermittent problems requires special attention -- much of your work is going to be an attempt to find ways to cause the problem to occur. Sometimes you may simply have to watch the net for long period and just wait.

Performance problems are often as hard to find as intermittent problems.

You will need at least a working knowledge of how networks work -- you can't really fix something unless you know its parts and how they interoperate.

Recognize the special characteristics of various network media. Ethernet comes in at least four distinct flavors (thick net, thin-net/cheapernet, 10-Base-T, and optical). Each of these has its own special failure modes. Switching and "secure" hubs have their own, sometimes devious, personalities.

Be sure you understand the topology of your network -- you will need to know what is connected to what, and know the device identity (e.g. the type of machine, its software, etc.) of each unit connected to the network.

You should almost always have a good notion what the results of any test should be before you run the test. In other words, you should how a test will come out on a well behaved network.

Don't assume that things are configured correctly -- most errors or problems are caused by user/pilot error. (Routing problems are the second most common kind of problem on TCP/IP networks.)

Don't make the mistake of assuming that your test equipment is running. It's almost always useful to make sure that your tools give you the results you expect when you attach to a working network.

Doing it

The following is a rough sequence of steps that I tend to follow when troubleshooting a TCP/IP problem. This framework should not be followed slavishly, rather it should be bent to accommodate the reality of the moment.

Gather evidence from the users.

Use users as your first line of testing -- make sure you understand their configuration and software base. *However, don't necessarily believe what users tell you -- they are not trained observers, nor do they necessarily use words the same way that you do.*

Good questions to ask are:

> "When was the last time you got something off the network?"
>
> "Are you experiencing a partial failure, or a total communications outage?"
>
> "Can you recreate the situation in which the problem occurs?"

If at all possible, go to the user's location and try it out yourself. (However, as will be mentioned below, immediately rushing off to the user's location is not always the best first step.)

Unless smoke is pouring out of the diskette slot, tell the user to leave his/her computer running.

Finding a good place to start.

You are going to need to run tests from somewhere. If you are like me, you might be tempted to run over to where the user happens to be and start there. It's a temptation to be resisted.

Rather, do as much as possible from your current location. You will have plenty of time later to run up stairwells, wait for elevators, brave city traffic, or drive to the "burbs".

With TCP/IP networks there are many problems where you do not need to be "on-site".

On the other hand, if you happen to be relatively close to one of the computers that the user is having trouble reaching, you might want to start there because it is right on the mainline path that is affecting the user.

Is there traffic?

Can you see traffic on the network? If not then perhaps there is something larger going on.

The easiest tools to see whether there is traffic are LEDs on hubs, network adapter cards, MAUs, and other devices of that nature. A packet monitor is also useful for this purpose, and will give you the additional bit of information whether the network is carrying the traffic you expect.

By "seeing" traffic, I don't necessarily mean that there is traffic on your LAN segment. Rather, I mean traffic anywhere in the network. For example, it is often useful to use SNMP to read interface counters from routes attached to the complaining user's LAN to see whether the routers are seeing any traffic.

You will want to see whether the user's computer is emitting packets when one would expect it to do so. LEDs are really useful here. If it isn't then there's probably a software or hardware problem inside the user's box.

Is there a there there? (Apologies to Gertrude Stein.)

TCP/IP networks often depend on logical names for devices. These are usually "domain names" from the Domain Name System (DNS). (The exception is when you are using something called "NIS".)

In many cases TCP/IP connectivity failures are caused not because of any problem in the mainline communications infrastructure. Rather, a computer's logical name may simply not be present in the naming databases.

You want to be sure that the target machine that the user is trying to reach is correctly listed in the naming system used by the user's machine. This means that you have to find out what mechanism the user's machine is using.

Many small networks depend on "host files" rather than the Domain Name System. Host files are simple text files which contain the pairings between host names and IP addresses. It's important that these files be consistent with one another. Otherwise you may find some users' computers working fine, others saying that a name can not be resolved, and others finding themselves being connected to the wrong resources.

So, one of the things you want to do early in your hunt is to see whether the both user's computer and the target that the user is trying to reach are both properly visible in the various name systems active on your network.

Exotic problems can arise if the Domain Name Servers are incomplete or inconsistent with one another or with people's host files. Many FTP servers, for example, refuse anonymous service unless the caller is completely and perfectly wired-in to the DNS system, including a reverse IP-address to host-name mapping (through the somewhat arcane "in-addr.arpa" domain.)

The best tools to find out what DNS is saying are either "nslookup", "dig", or "host". Except for the version found on FTP Software's PC/TCP and OnNet products, these tools are mainly found on Unix systems. None of these tools are particularly easy to use to their fullest extent unless you have a good understanding how the DNS servers interact with one another and how the various types of DNS records work. But in the TCP/IP world, this is an important skill.

(Recently I've started to see a number of reports of intermittent name lookup problems from people using World Wide Web browsers [such as those from Netscape or Spry.] I've often found these not to be really name lookup problems at all. Rather people were using PPP or SLIP dial-up links and the links were dropping.)

Can you get from "here" to "there" and back again?

The basic job of any network is to get packets from one place to another. In particular, you want to be able to make sure that packets from the user's computer are reaching the intended destination. And, what many people forget, you also need to ensure that packets from the intended destination are, in fact, making it back to the user's computer.

The tools to do this are "ping" and "traceroute".

You want to use ping to bounce a packet off of the target computer. And traceroute tells you the path(s) that your packets are taking and, if they are failing to make it, where they seem to be being lost.

Ping and traceroute are really incredibly useful tools. With neither muss nor fuss, ping tests basic connectivity, while traceroute can tell you everything from the sequence of routers traversed to the portion of overall transit latency added by each router-to-router hop. One thing which traceroute will show you that few other tools can are "black holes" or routing loops. These are places where two or more routers are each referring to one another as a better path. Once a packet enters, it probably will never come out.

To use these tools, you need to be somewhat close (in a network topology sense) to where the user or server lies. This is where I find a small 10-Base-T hub to be a handy tool -- I disconnect the user's computer from the net, attach the hub in the user's place, and then plug both the user's machine and my test gear into the hub.

You might also want to try to start from the target computer and bounce some packets off of the user's computer. The trouble with this, however, is that some of the TCP/IP packages for PC's are, euphemistically speaking, seriously "TCP/IP challenged" and either do not respond to ping requests or do not do so unless in a receptive state. This is particularly true of TCP/IP stacks which depend on DLL's rather than VxDs or TSRs.

Unfortunately, few of the popular commercial TCP/IP stacks have a traceroute applications, and I have yet to find one that is correctly implemented. And many Windows Sockets (WinSock) implementations are inadequate to support either ping or traceroute.

Assuming that you can not get data across the network, the first thing to look at is whether the user's packets are being properly routed.

The best way to handle this is first to ping a computer which is on the same LAN segment as the user's machine. If this does not work, take a very close look at the configuration of the user's machine, especially its IP address and subnet mask.

If local machines can be pinged, then see if you can ping the local router's near side interface.

You're probably asking "near side interface"? What's that? In the TCP/IP world, IP addresses are not associated with computers, but rather with their interfaces to the network. A router will almost certainly have a different IP address on each of its ports. The "near side interface" is the one on the router which is attached to your LAN segment. You might have to ask your network administrator for this address, or you might get it from a machine which is properly working, or you might simply monitor the various forms of IP routing traffic.

Once you know that you can ping the nearest router, you need to see whether the user's computer is, in fact, configured to use that router to handle off-subnet traffic. Take a very close look at the user's IP address, subnet mask, and "default router" setting, if any. If there is no default router set, make sure that the user's machine is listening for routing protocols (usually the Routing Information Protocol, or RIP for short) and also check that the router is emitting RIP.

Sometimes user machines use a protocol called "Router Discovery Protocol" or RDP to locate routers, but relatively few sites are using it yet.

Is there noise or selective data loss?

TCP/IP can be reasonably insensitive to noise and variations in network transit delays. Some implementations are better than others. Nevertheless, there are times when the noise on a link is simply too much to bear.

The easiest way to test noise is to send a few dozen pings back and forth. You should not lose more than a small percentage.

Toolkit

What sort of tools do you need to troubleshoot TCP/IP networks? There are many, many tools. In fact there is an entire catalogue of such tools available: RFC 1470 "FYI on a Network Management Tool Catalog, Tools for Monitoring and Debugging TCP/IP Internets and Interconnected Devices"

Here's some of the general kinds of tools you are going to need

- Ping -- "Ping" is almost the ultimate network test tool. "Ping" simply sends an echo request packet to another computer on the network and that other computer (hopefully) answers. You want a really flexible ping implementation, however, that is able to control the rate and size of echo request packets.

- Traceroute -- Close on the heels of "ping" is "traceroute". This tool is used to create a map of the path your packets take as they move from wherever you are to wherever you are trying to reach. Although the output of traceroute can be somewhat difficult to interpret at times, it is an absolutely essential tool.

- Packet monitor -- You will want a way to view and decode the packets flowing on your network. There are a large number of so-called "protocol analyzers" which can do the job. These include the Network General Sniffer, FTP Software's LANwatch, or the freely available tcpdump.

- For Domain Name System probing the tool you need will be named either "nslookup", "dig" or "host". Unfortunately, these are rare on non-Unix systems.

- You will want a simple SNMP MIB browser. You do not want a full SNMP management station.

Unless you have a herd of elephants handy to carry various workstations and have a budget to match and the patience to reconfigure them whenever you set-up a troubleshooting listening post, you are going to have to be somewhat selective about which tools you actually use on any particular occasion.

Whenever I've set forth on a troubleshooting adventure, I've tried to treat it like a backpacking trip -- take only the most flexible and lightweight tools. Among the tools I would take, if I had the opportunity, are the following:

- A notebook PC equipped with a good strong battery, a PCMCIA Ethernet card (i.e. one that is able to handle promiscuous mode) and the following software:

  - ◊ FTP Software's LANWatch, Network General's Sniffer, or any other good packet monitor.

  - ◊ An all-in-one package such as Empirical Tools and Technologies' Dr. Watson, the Network Detective's Assistant. (I'm somewhat biased in favor of this product; the concept and form were developed during my years at Epilogue Technology Corporation. I reduced a portion of these ideas to software at Empirical. LAN Magazine awarded it the 1994 Product of the Year for troubleshooting.)

  - ◊ A TCP/IP stack with a rich set of applications. I use FTP Software's PC/TCP, OnNet package for this. It has a fairly nice set of testing applications (such as the previously mentioned "host" program) in addition to the basic "telnet" and "ftp" functions found in other TCP/IP packages. And sometimes it is extremely valuable to perform a "telnet" into a non-standard TCP port. There are some other TCP/IP stacks which I would not recommend for troubleshooting because they are lacking in basic functions such as IP address to hostname lookup capabilities.

  - ◊ Simple SNMP MIB browser.

- A combination protocol and cable test device such as a Fluke Model 675 Lanmeter.

- A small toolpouch with small screwdrivers, wirestripper, a small flashlight, cable ties, marking pen, pencil and paper.

- A handful of adapter hardware and cables -- a small 10-Base-T hub, RJ45 barrel connectors, 10-Base-T patch and "null modem" cables, 10-Base-T/AUI MAU, terminators, tees, etc.