15. MAINTENANCE AND TROUBLESHOOTING

1. Introduction

How you establish the support infrastructure for your network is as important as what type of equipment you use. Unlike wired connections, problems with a wireless network are often invisible, and can require more skill and more time to diagnose and remedy. Interference, wind, and new physical obstructions can cause a long-running network to fail. This chapter details a series of strategies to help you build a team that can support your network effectively.
We also describe some standard troubleshooting techniques that have proven successful in solving problems in networks in general.

2. Building your team

Every village, company or family has individuals who are intrigued by technology.
They are the ones found splicing the television cable, re- wiring a broken television or welding a new piece to a bicycle. These people will take interest in your network and want to learn as much about it as possible. Though these people are invaluable resources, you must avoid imparting all of the specialised knowledge of wireless networking to only one person.
If your only specialist loses interest or finds better paying work somewhere else, they take the knowledge with them when they go.
There may also be many ambitious young adults who will be interested and have the time to listen, help, and learn about the network.
Again, they are very helpful and will learn quickly, but the project team must focus their attention on those who are best placed to support the network in the coming months and years.
Young adults and teenagers will go off to university or find employment taking their knowledge and skills about the network with them.

These youngsters may also have little influence in the community, where

an older individual is likely to be more well-known and less impulsive regarding making decisions that affect the network as a whole.
Even though these individuals might have less time to learn and might appear to be less interested, their involvement and proper education about the system can be critical.
Therefore, a key strategy in building a support team is to balance and to distribute the knowledge among those that are best placed to support the network for the long term.
You should involve young people, but alongside them involve a mix of more mature contributors. Find people who are committed to the community, who have roots in the community, who are motivated to learn and teach. A complementary strategy is to divide up the duties and functions, and to document all methodology and procedures. In this way, people can be trained easily, and substituted with little effort.
On one project the training team selected a bright young university graduate who had returned to his village. He was very motivated and learned quickly. Soon he was an expert in IT and networking skills as well as the local network. He was able to deal with many different problems, from fixing a PC to rewiring Ethernet cable. Unfortunately, two months after the project launch he was offered a government job and left the community.
Even a better salary could not keep him, since the prospect of a stable government job was too appealing. All of the knowledge about the network and how to support it left with him. The training team had to return and begin the training again, this time with local people who were known to be remaining in the village. Although this retraining took much longer, the community could be assured that the knowledge and skills would remain in the village for a longer time.

It is often best to find a local partner organisation or a local manager, and work with them to find the right technical team. Values, history, local politics, and many other factors will be important to them, while remaining completely unfathomable to people who are not from that community. The best approach is to coach your local partner, providing them with sound criteria, make sure that they understand that criteria, and set firm boundaries. Such boundaries should include rules about nepotism and patronage, considering, of course, your own local situation. It may be impossible to say that you cannot hire kin, but it is best to provide a means of checks and balances.

Where a candidate is kin, there should be clear criteria and a second authority in deciding upon their candidacy.
It is also important that the local partner is given authority and is not undermined by the project organisers, thus compromising their ability to manage. They will be best able to judge who will work best with them.
If they are well educated in this process, then your requirements should be satisfied. Troubleshooting and support of technology is an abstract art.
The first time you look at an abstract painting, it may just look to you like a bunch of random paint splatters. After reflecting on the composition for a time, you may come to appreciate the work as a whole, and the "invisible" coherence becomes very real.
The neophyte looking at a wireless network may see the antennas and wires and computers, but it can take a while for them to appreciate the point of the "invisible" network. In rural areas, it can often take a huge leap of understanding before locals will appreciate an invisible network that is simply dropped into their village. Therefore, a phased approach is needed to ease people into supporting technology systems. Again, the best method is involvement.
Once the participants are chosen and committed to the project, involve them as much as possible. Let them "drive". Give them the cable crimper or keyboard and show them how to do the work. Even if you do not have time to explain every detail and even if it will take longer, they need to be involved physically and see not only what has been done, but how much work was done.
The scientific method is taught in many schools and universities.
Many people learn about it by the time they reach high-school science class. Simply put, answers are achieved and problems solved by taking a set of possibilities, then slowly eliminating them through binary tests until you are left with one or only a few possibilities.
With those possibilities in mind, you complete the experiment.
You then test to see if the experiment yields something similar to the expected result. If it did not, you re-calculate your expected result and try again. A young person you might be thinking of employing in your project may have been introduced to the concept, but likely will not have had the opportunity to troubleshoot complex problems.
Even if they are familiar with the scientific method, they might not think to apply it to resolving real problems. This method is very effective, although time consuming. It can be sped up by making logical assumptions. For example, if a long-running access point suddenly stops

working after a storm, you might suspect a power supply related problem and thus skip most of the procedure.
People charged with supporting technology should be taught how to troubleshoot using this method, as there will be times when the problem is neither known nor evident. Simple decision trees or flow charts can be made that test variables, and then elimination of the variables to isolate the problem can be suggested.
Of course, these charts should not be followed blindly.
It is often easier to teach this method using a non technological problem first. For example, have your student develop a problem resolution procedure on something simple and familiar, like a battery powered television. Start by sabotaging the television. Give them a battery that is not charged. Disconnect the aerial. Insert a broken fuse. Test the student, making it clear that each problem will show specific symptoms, and point the way as to how to proceed. Once they have fixed the television, have them apply this procedure to a more complicated problem. In a network, you can change an IP address, switch or damage cables, use the wrong SSID, or orient the antenna in the wrong direction. It is important that they develop a methodology and procedure to resolve these problems.

3. Proper troubleshooting techniques

No troubleshooting methodology can completely cover all problems you will encounter when working with wireless networks. But often, problems come down to one of a few common mistakes. Here are a few simple points to keep in mind that can get your troubleshooting effort working in the right direction.

- Don't panic. If you are troubleshooting a system, it means that it was working at one time, probably very recently. Before jumping in and making changes, survey the scene and assess exactly what is broken. If you have historical logs or statistics to work from, all the better. If others were using it before it started having problems, ask them to help you by having them tell you what was happening before it stopped working. Be thorough, but don't make it sound like you are accusing them of breaking it.
  They may have important information that will help you fix things and you want them to be on your side. Be sure to collect information first, so you can make an informed decision before

making changes.

- **Make a backup**. This applies before you notice problems, as well as after. If you make a complicated software change to a system, having a backup means that you can quickly restore it to the previous settings and start again. When troubleshooting very complex problems, having a configuration that "sort-of" works can be much better than having a mess that doesnt work at all (and that you cant easily restore from memory).
  Even in a broken configuration, make a backup copy of the parts of the system you will be changing before you try to make significant changes. If your changes result in an even worse state than when you first started working on it, you will at least have a known situation to go back to.

- **Is it plugged in?** This step is often overlooked until many other avenues are explored. Plugs can be accidentally (or intentionally) unplugged very easily. Is the lead connected to a good power source? Is the other end connected to your device? Is the power light on? It may sound silly, but you will feel even sillier if you spend a lot of time checking out an antenna feed line only to realise that the AP was unplugged the entire time. Trust me, it happens more often than most of us would care to admit.

- **What was the last thing changed?** If you are the only person with access to the system, what is the last change you made? If others have access to it, what is the last change they made and when? When was the last time the system worked? Often, system changes have unintended consequences that may not be immediately noticed. Roll back that change and see what effect it has on the problem.

- **Look at date/timestamps on files.** Every file on a modern computer system has a date & time associated with it showing when it was created or last changed. On a properly running system, most of the system files will have date/timestamps from months or even years ago. If the system or network was running fine until an hour or so ago, files which have a timestamp within the past few minutes to an hour ago could provide clues about

what changed.

- **The known good.** This idea applies to hardware, as well as software. A known good is any component that you can replace in a complex system to verify that its counterpart is in good, working condition. For example, you may carry a tested Ethernet cable in a tool kit. If you suspect problems with a cable in the field, you can easily swap out the suspect cable with the known good and see if things improve. This is much faster and less error-prone than re-crimping a cable, and immediately tells you if the change fixes the problem. Likewise, you may also pack a backup battery, antenna cable, or a CD-ROM with a known good configuration for the system. When fixing complicated problems, saving your work at a given point lets you return to it as a known good, even if the problem is not yet completely solved.

- **Determine what still works.** This will help you "put a fence around the problem". While complex systems like a wireless network can be made up of many different components, it is likely that the problem is only with a very small number of them. If, for example, somebody in a lab complains they can't access the Internet, check to see if others in the same lab are experiencing the same problem. Is there connectivity in another lab or elsewhere in the building? If the problem is just with one user or within one room, you would want to concentrate your efforts on the equipment in just that one space. If the outage were more widespread, perhaps looking at the equipment where your outside connections come in is more appropriate.

- **Do no harm.** If you don't fully understand how a system works, don't be afraid to call in an expert. If you are not sure if a particular change will damage another part of the system, then either find someone with more experience to help you or devise a way to test your change without doing damage. Putting a penny in place of a fuse may solve the immediate problem, but it may also burn down the building.

Your troubleshooting team will need to have good troubleshooting skills, but may not be competent enough to configure a router from scratch or

crimp a piece of LMR-400.

It is often much more efficient to have a number of backup components on-hand, and train your team to be able to swap out the entire broken part. This could mean having an access point or router pre-configured and sitting in a locked cabinet, plainly labelled and stored with backup cables and power supplies. Your team can swap out the failed component, and either send the broken part to an expert for repair, or arrange to have another backup sent in.
Assuming that the backups are kept secure and are replaced when used, this can save a lot of time for everyone.

4. Common network problems

Now we will take a look at some common network problems that you are almost certain to face.

Often, connectivity problems come from failed components, adverse weather, or simple misconfiguration.

Once your network is connected to the Internet or opened up to the general public, considerable threats will come from the network users themselves.

These threats can range from the benign to the outright malevolent, but all will have impact on your network if it is not properly configured. This section looks at some common problems found once your network is in use.

Locally hosted websites

If a university hosts its website locally, visitors to the website from outside the campus and the rest of the world will compete with the university's staff for Internet bandwidth.
This includes automated access from search engines that periodically spider your entire site.

One solution to this problem is to use split DNS and mirroring.
The university mirrors a copy of its websites to a server at, say, a European hosting company, and uses split DNS to direct all users from

outside the university network to the mirror site, while users on the university network access the same site locally.

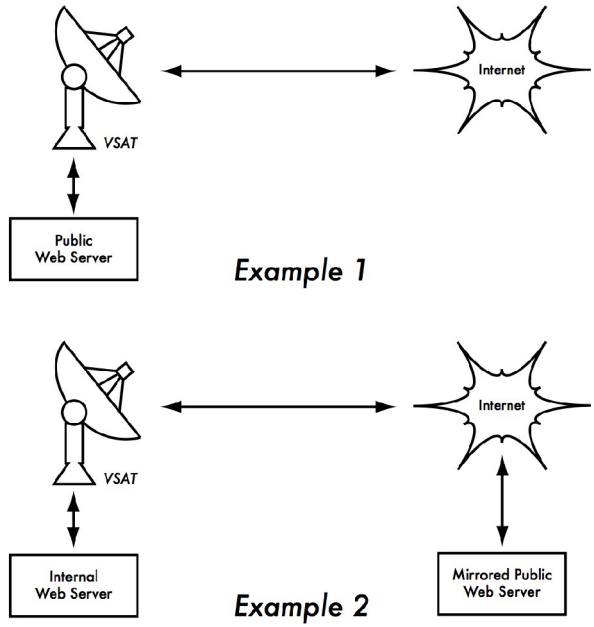

Figure MT 1: In Example 1, all website traffic coming from the Internet must traverse the VSAT. In Example 2, the public web site is hosted on a fast European service, while a copy is kept on an internal server for very fast local access. This improves the VSAT connection and reduces load times for web site users.

Open proxies

A proxy server should be configured to accept only connections from the university network, not from the rest of the Internet.

This is because people elsewhere will connect and use open proxies for a variety of reasons, such as to avoid paying for international bandwidth. The way to configure this depends on the proxy server you are using. For example, you can specify the IP address range of the campus network in your squid.conf file as the only network that can use Squid.

Alternatively, if your proxy server lies behind a border firewall, you can configure the firewall to only allow internal hosts to connect to the proxy port.

Open relay hosts

An incorrectly configured mail server will be found by unscrupulous people on the Internet, and be used as a relay host to send bulk email and spam.
They do this to hide the true source of the spam, and avoid getting caught. To test for an open relay host, the following test should be carried out on your mail server (or on the SMTP server that acts as a relay host on the perimeter of the campus network). Use telnet to open a connection to port 25 of the server in question.

telnet mail.uzz.ac.zz 25

Then, if an interactive command-line conversation can take place (for example, as follows), the server is an open relay host:

MAIL FROM: spammer@waste.com
250 OK - mail from
RCPT TO: innocent@university.ac.zz
250 OK - rcpt to spammer@waste.com

Instead, the reply after the first MAIL FROM should be something like:

550 Relaying is prohibited.

An online tester is available at sites such as
http://www.mailradar.com/openrelay/ or http://www.checkor.com/

Since bulk emailers have automated methods to find such open relay hosts, an institution that does not protect its mail systems is almost guaranteed to be found and abused.

Configuring the mail server not to be an open relay consists of specifying the networks and hosts that are allowed to relay mail through them in the MTA (eg., Sendmail, Postfix, Exim, or Exchange).
This will likely be the IP address range of the campus network.

Peer-to-peer networking

Bandwidth abuse through peer-to-peer (P2P) file-sharing programs can be prevented in the following way.
Make it impossible to install new programs on campus computers. By not giving regular users administrative access to PC workstations, it is possible to prevent the installation of bandwidth hungry applications. Many institutions also standardise on a desktop build, where they install the required operating system on one PC. They then install all the necessary applications on it, and configure these in an optimal way. The PC is also configured in a way that prevents users from installing new applications. A disk image of this PC is then cloned to all other PCs using software such as Partition Image (see http://www.partimage.org/) or Drive Image Pro (see http://www.powerquest.com/).

From time to time, users may succeed in installing new software or otherwise damaging the software on the computer (causing it to hang often, for example). When this happens, an administrator can simply put the disk image back, causing the operating system and all software on the computer to be exactly as specified.

Programs that install themselves (from the Internet)

There are programs that automatically install themselves and then keep on using bandwidth.
Some programs are spyware, which keep sending information about a user's browsing habits to a company somewhere on the Internet.

These programs are preventable to some extent by user education and locking down PCs to prevent administrative access for normal users. In other cases, there are software solutions to find and remove these problem programs, such as Spychecker (http://www.spychecker.com/).

Windows updates

Microsoft Windows operating systems assume that a computer with a LAN connection has a good link to the Internet, and automatically downloads security patches, bug fixes and feature enhancements from the Microsoft Web site. This can consume massive amounts of bandwidth on an expensive Internet link.

The two possible approaches to this problem are:

- Disable Windows updates on all workstation PCs. The security updates are very important for servers, but whether workstations in a protected private network such as a campus network need them is debatable.

- Install a Software Update Server. This is a free program from Microsoft that enables you to download all the updates from Microsoft overnight on to a local server and distribute the updates to client workstations from there. In this way, Windows updates need not use any bandwidth on the Internet link during the day. Unfortunately, all client PCs need to be configured to use the Software Update Server for this to have an effect.
  If you have a flexible DNS server, you can also configure it to answer requests for windowsupdate.microsoft.com and direct the updater to your update server. This is only a good option for large networks, but can save untold amounts of Internet bandwidth.

Programs that assume a high bandwidth link

In addition to Windows updates, many other programs and services assume that bandwidth is not a problem, and therefore consume bandwidth for reasons the user might not predict. For example, anti-virus packages (such as Norton AntiVirus) periodically update themselves automatically and directly from the Internet. It is better if these updates are distributed from a local server.
Other programs, such as the RealNetworks video player, automatically download updates and advertisements, as well as upload usage patterns back to a site on the Internet. Innocuous looking applets (like Konfabulator and Dashboard widgets) continually poll Internet hosts for updated information. These can be low bandwidth requests (like weather or news updates), or very high bandwidth requests (such as webcams). These applications may need to be throttled or blocked altogether.
The latest versions of Windows and Mac OS X also have a time synchronisation service. This keeps the computer clock accurate by connecting to time servers on the Internet. It is more efficient to install a local time server and distribute accurate time from there, rather than to tie up the Internet link with these requests.

Worms and viruses

Worms and viruses can generate enormous amounts of traffic. It is therefore essential that anti-virus protection is installed on all PCs. Furthermore, user education about executing attachments and responding to unsolicited email is essential. In fact, it should be a policy that no workstation or server should run unused services.
A PC should not have shares unless it is a file server; and a server should not run unnecessary services either.
For example, Windows and Unix servers typically run a web server service by default. This should be disabled if that server has a different function; the fewer services a computer runs, the less there is to exploit.

Email forwarding loops

Occasionally, a single user making a mistake can cause a problem. For example, a user whose university account is configured to forward all mail to her Yahoo account.

The user goes on holiday. All emails sent to her in her absence are still forwarded to her Yahoo account, which can grow to only 2 MB.
When the Yahoo account becomes full, it starts bouncing the emails back to the university account, which immediately forwards it back to the Yahoo account.
An email loop is formed that might send hundreds of thousands of emails back and forth, generating massive traffic and crashing mail servers.

There are features of mail server programs that can recognise loops. These should be turned on by default.
Administrators must also take care that they do not turn this feature off by mistake, or install an SMTP forwarder that modifies mail headers in such a way that the mail server does not recognise the mail loop.

Large downloads

A user may start several simultaneous downloads, or download large files such as 650MB ISO images.
In this way, a single user can use up most of the bandwidth.
The solutions to this kind of problem lie in training, offline downloading, and monitoring.

Offline downloading can be implemented in at least two ways:

- At the University of Moratuwa, a system was implemented using URL redirection. Users accessing ftp:// URLs are served a directory listing in which each file has two links: one for normal downloading, and the other for offline downloading. If the offline link is selected, the specified file is queued for later download and the user notified by email when the download is complete. The system keeps a cache of recently downloaded files, and retrieves such files immediately when requested again. The download queue is sorted by file size. Therefore, small files are downloaded first. As some bandwidth is allocated to this system even during peak hours, users requesting small files may receive them within minutes, sometimes even faster than an online download.

- Another approach would be to create a web interface where users enter the URL of the file they want to download. This is then downloaded overnight using a cron job or scheduled task. This system would only work for users who are not impatient, and are familiar with what file sizes would be problematic for download during the working day.

Sending large files

When users need to transfer large files to collaborators elsewhere on the Internet, they should be shown how to schedule the upload. In Windows, an upload to a remote FTP server can be done using an FTP script file, which is a text file containing FTP commands.

Users sending each other files

Users often need to send each other large files. It is a waste of bandwidth to send these via the Internet if the recipient is local. A file share should be created on the local Windows/Samba/Mac web server, where a user can put the large file for others to access.
Alternatively, a web front-end can be used for a local web server to accept a large file and place it in a download area. After uploading it to the web server, the user receives a URL for the file.
He can then give that URL to his local or international collaborators, and when they access that URL they can download it.

This is what the University of Bristol has done with their FLUFF system. The University offers a facility for the upload of large files (FLUFF) available from *http://www.bris.ac.uk/it-services/applications/fluff/*

There are also tools like SparkleShare (*http://sparkleshare.org/)* and LipSync (*https://github.com/philcryer/lipsync*) which are Open Source packages that you can install and set up yourself to do much the same thing.
There are also new online free services such as Google Drive which can be setup for file sharing and common editing.

Consider using rsync (http://rsync.samba.org/) for those users who need to send each other the same/similar large files on a regular basis.
The Rsync protocol is a synchronisation rather than a straightforward file transfer protocol.
Rather than simply sending a file from start to end, it checks with an rsync server on the destination host to see if the file being sent already exists.
If it does, both sides compare their copies and the sender transmits over only the differences to the destination.

For example, if a 10 MB database of research data only has 23 KB of new data vs. the last version, only the 23 KB of changes will be transmitted.
Rsync can also use the SSH protocol, providing a secure transport layer for sync actions.

5. Trouble tracking and reporting

Troubleshooting is only one half of the task of problem-solving on a wireless network.
Once a problem has been diagnosed and fixed, it needs to be documented in a permanent way so that others who work on the network either now or in the future, will be able to learn from the incident.

Keeping a record of problems and incidents that occur is also a good way to track and fix long-term problems that may occur, say, once every few months but follow a definite pattern.

You can also reduce the complexity and frustration of troubleshooting a problem if you keep a log of every change made to the network.

The Logbook is where you and your team write down every change made to a system, along with the date and time the change was made. For example:

23 July 10:15AM Changed default route on host alpha from 123.45.67.89 to 123.56.78.1 because upstream ISP moved our gateway.

As your network grows, consider installing a trouble-tracking system like JIRA or Bugzilla to help better keep track of who is working on what problem and what happened during that work. This provides a history of what was worked on and how it was fixed and also provides an orderly method for assigning tasks and helping to prevent things like two people stepping on each other's work while they both try to fix the same problem.

Trouble-ticketing systems are a subject that would fill an entire separate book, so we will only mention them briefly here to make you aware of them. They can also be quite complex to set up, so for simple networks, a logbook will do.