3
Network Design

Before purchasing equipment or deciding on a hardware platform, you should have a clear idea of the nature of your communications problem. Most likely, you are reading this book because you need to connect computer networks together in order to share resources and ultimately reach the larger global Internet. The network design you choose to implement should fit the communications problem you are trying to solve. Do you need to connect a remote site to an Internet connection in the center of your campus? Will your network likely grow to include several remote sites? Will most of your network components be installed in fixed locations, or will your network expand to include hundreds of roaming laptops and other devices?

When solving a complex problem, it is often useful to draw a picture of your resources and problems. In this chapter, we will look at how other people have built wireless networks to solve their communication problems, including diagrams of the essential network structure. We will then cover the networking concepts that define TCP/IP, the primary networking language currently spoken on the Internet. We will then demonstrate several common methods for getting your information to flow efficiently through your network and on to the rest of the world.

Designing the physical network

It may seem odd to talk about the "physical" network when building wireless networks. After all, where is the physical part of the network? In wireless networks, the physical medium we use for communication is obviously electromagnetic energy. But in the context of this chapter, the physical network refers to the mundane topic of where to put things. How do you arrange the equipment so that you can reach your wireless clients? Whether

they fill an office building or stretch across many miles, wireless networks are naturally arranged in these three logical configurations:

- Point-to-point links
- Point-to-multipoint links
- Multipoint-to-multipoint clouds

The physical network layout you choose will depend on the nature of the problem you are trying to solve. While different parts of your network can take advantage of all three of these configurations, any individual link will fall into one of the above topologies. The application of each of these topologies is best described by example.

Point-to-point

Point-to-point links typically provide an Internet connection where such access isn't otherwise available. One side of a point-to-point link will have an Internet connection, while the other uses the link to reach the Internet. For example, a university may have a fast frame relay or VSAT connection in the middle of campus, but cannot afford such a connection for an important building just off campus. If the main building has an unobstructed view of the remote site, a point-to-point connection can be used to link the two together. This can augment or even replace existing dial-up links. With proper antennas and clear line of sight, reliable point-to-point links in excess of thirty kilometers are possible.

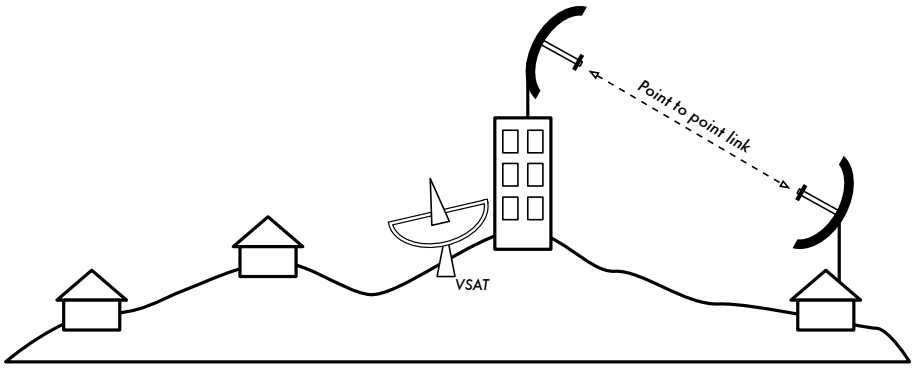

Figure 3.1: A point-to-point link allows a remote site to share a central Internet connection.

Of course, once a single point-to-point connection has been made, more can be used to extend the network even further. If the remote building in our

example is at the top of a tall hill, it may be able to see other important locations that can't be seen directly from the central campus. By installing another point-to-point link at the remote site, another node can join the network and make use of the central Internet connection.

Point-to-point links don't necessarily have to involve Internet access. Suppose you have to physically drive to a remote weather monitoring station, high in the hills, in order to collect the data which it records over time. You could connect the site with a point-to-point link, allowing data collection and monitoring to happen in realtime, without the need to actually travel to the site. Wireless networks can provide enough bandwidth to carry large amounts of data (including audio and video) between any two points that have a connection to each other, even if there is no direct connection to the Internet.

Point-to-multipoint

The next most commonly encountered network layout is ***point-to-multipoint***. Whenever several nodes[1] are talking to a central point of access, this is a point-to-multipoint application. The typical example of a point-to-multipoint layout is the use of a wireless access point that provides a connection to several laptops. The laptops do not communicate with each other directly, but must be in range of the access point in order to use the network.

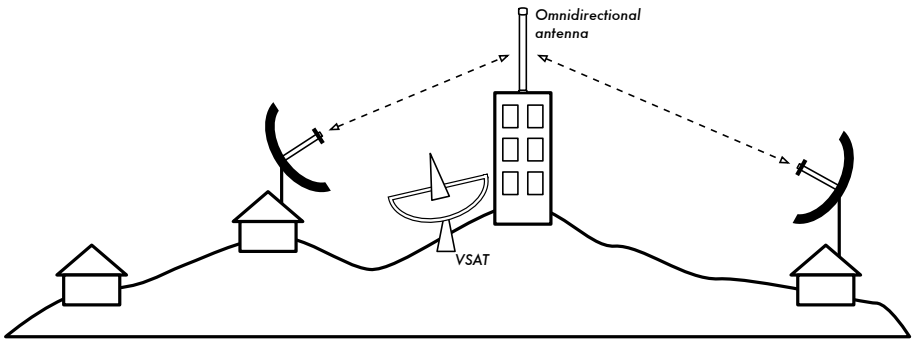

Figure 3.2: The central VSAT is now shared by multiple remote sites. All three sites can also communicate directly at speeds much faster than VSAT.

Point-to-multipoint networking can also apply to our earlier example at the university. Suppose the remote building on top of the hill is connected to the central campus with a point-to-point link. Rather than setting up several point-to-point links to distribute the Internet connection, a single antenna

---

1. A ***node*** is any device capable of sending and receiving data on a network. Access points, routers, computers, and laptops are all examples of nodes.

could be used that is visible from several remote buildings. This is a classic example of a wide area ***point*** (remote site on the hill) ***to multipoint*** (many buildings in the valley below) connection.

Note that there are a number of performance issues with using point-to-multipoint over very long distance, which will be addressed later in this chapter. Such links are possible and useful in many circumstances, but don't make the classic mistake of installing a single high powered radio tower in the middle of town and expecting to be able to serve thousands of clients, as you would with an FM radio station. As we will see, data networks behave very differently than broadcast radio.

Multipoint-to-multipoint

The third type of network layout is ***multipoint-to-multipoint***, which is also referred to as an ***ad-hoc*** or ***mesh*** network. In a multipoint-to-multipoint network, there is no central authority. Every node on the network carries the traffic of every other as needed, and all nodes communicate with each other directly.

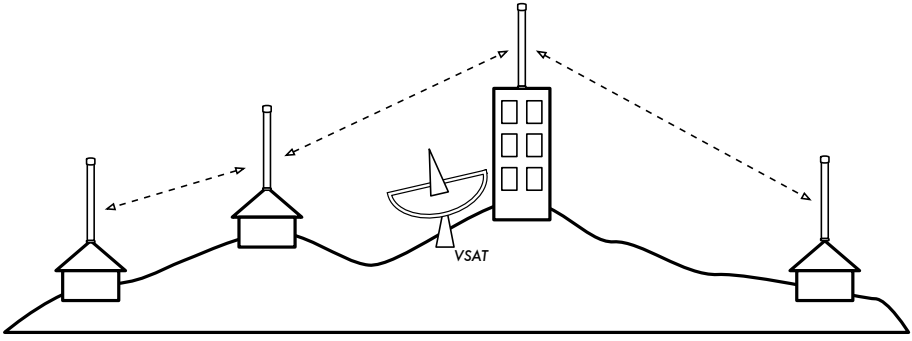

Figure 3.3: A multipoint-to-multipoint mesh. Every point can reach each other at very high speed, or use the central VSAT connection to reach the Internet.

The benefit of this network layout is that even if none of the nodes are in range of a central access point, they can still communicate with each other. Good mesh network implementations are self-healing, in that they automatically detect routing problems and fix them as needed. Extending a mesh network is as simple as adding more nodes. If one of the nodes in the "cloud" happens to be an Internet gateway, then that connection can be shared among all of the clients.

Two big disadvantages to this topology are increased complexity and lower performance. Security in such a network is also a concern, since every participant potentially carries the traffic of every other. Multipoint-to-multipoint networks tend to be complicated to troubleshoot, due to the large number of

changing variables as nodes move around. Multipoint-to-multipoint clouds typically do not have the same capacity as point-to-point or point-to-multipoint networks, due to the additional overhead of managing the network routing and increased contention in the radio spectrum.

Nevertheless, mesh networks are useful in many circumstances. We will see an example of how to build a multipoint-to-multipoint mesh network using a routing protocol called OLSR at the end of this chapter.

Use the technology that fits

All of these network designs can be used to complement each other in a large network, and can obviously make use of traditional wired networking techniques whenever possible. It is a common practice, for example, to use a long distance wireless link to provide Internet access to a remote location, and then set up an access point on the remote side to provide local access. One of the clients to this access point may also act as a mesh node, allowing the network to spread organically between laptop users who all ultimately use the original point-to-point link to access the Internet.

Now that we have a clear idea of the way that wireless networks are typically arranged, we can begin to understand how communication is possible over such networks.

The logical network

Communication is only possible when the participants speak a common language. But once the communication becomes more complex than a simple ongoing broadcast, ***protocol*** becomes just as important as language. All of the people in an auditorium may speak English, but without a set of rules in place to establish who has the right to use the microphone, the communication of an individual's ideas to the entire room is nearly impossible. Now imagine an auditorium as big as the world, full of all of the computers that exist. Without a common set of communication protocols to regulate when and how each computer can speak, the Internet would be a chaotic mess where every machine tries to speak at once.

TCP/IP refers to the suite of protocols that permit conversations to happen on the global Internet. By understanding TCP/IP, you can build networks that will scale to virtually any size, and will ultimately become part of the global Internet.

The TCP/IP model

Data networks are often described as being built on many layers. Each layer depends on the operation of all of the underlying layers before communication can take place, but only needs to exchange data with the layer above or beneath it. The TCP/IP model of networking[2] describes five layers, as shown in this diagram:

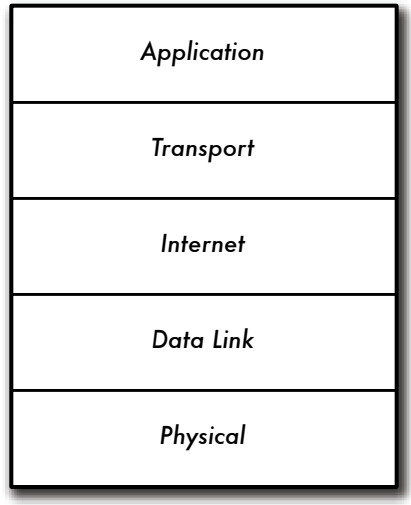

Figure 3.4: The TCP/IP networking model.

The previous section on network layouts described layer one: the ***physical layer***. This is the physical medium over which communications take place. This can be a copper CAT5 cable, a fiber optic bundle, radio waves, or just about any other medium.

The next layer up is referred to as the ***data link layer***. Whenever two or more nodes share the same physical medium (for example, several computers plugged into a hub, or a room full of laptops all using the same radio channel) they use the data link layer to determine whose turn it is to transmit on the medium. Common examples of data link protocols are Ethernet, Token Ring, ATM, and the wireless networking protocols (802.11a/b/g). Communication on this layer is said to be ***link local***, since all nodes connected at this layer can communicate with each other directly. On networks modeled after Ethernet, nodes are referred to by their ***MAC address***, which is a unique 48 bit number assigned to every networking device when it is manufactured.

---

2. The TCP/IP model isn't an international standard, and its definition varies. It is included here as a pragmatic model used for understanding and troubleshooting Internet networks.

Just above the data link layer is the ***Internet layer***. For TCP/IP, this is the Internet Protocol (***IP***). At the Internet layer, packets can leave the link local network and be retransmitted on other networks. Routers perform this function on a network by having at least two network interfaces, one on each of the networks to be interconnected. Nodes on the Internet are reached by their globally unique ***IP address***.

Once Internet routing is possible, a method is needed to reach a particular service at a given IP address. This function is filled by the next layer, the ***transport layer***. TCP and UDP are common examples of transport layer protocols. Some protocols at the transport layer (such as TCP) ensure that all of the data has arrived at the destination, and is reassembled and delivered to the next layer in the proper order.

Finally, at the top of the pile we have the ***application layer***. This is the layer that most network users are exposed to, and is the level at which human communication happens. HTTP, FTP, and SMTP are all application layer protocols. The human sits at the top of all of the layers, and needs little or no knowledge of the layers beneath to effectively use the network.

One way to look at the TCP/IP model is to think of a person delivering a letter to an office building downtown. They first need to interact with the road itself (the physical layer), pay attention to other traffic on the road (the data link layer), turn at the proper place to connect to other roads and arrive at the correct address (the Internet layer), go to the proper floor and room number (the transport layer), and finally find the recipient or a receptionist who can take the letter from there (the application layer). The five layers can be easily remembered by using the mnemonic "**P**lease **D**on't **L**ook **I**n **T**he **A**ttic," which of course stands for "**P**hysical / **D**ata **L**ink / **I**nternet / **T**ransport / **A**pplication."

802.11 wireless networks

Before packets can be forwarded and routed to the Internet, layers one (the physical) and two (the data link) need to be connected. Without link local connectivity, network nodes cannot talk to each other and route packets.

To provide physical connectivity, wireless network devices must operate in the same part of the radio spectrum. As we saw in chapter two, this means that 802.11a radios will talk to 802.11a radios at around 5GHz, and 802.11b/g radios will talk to other 802.11b/g radios at around 2.4GHz. But an 802.11a device cannot interoperate with an 802.11b/g device, since they use completely different parts of the electromagnetic spectrum.

More specifically, wireless cards must agree on a common channel. If one 802.11b radio card is set to channel 2 while another is set to channel 11, then the radios cannot communicate with each other.

When two wireless cards are configured to use the same protocol on the same radio channel, then they are ready to negotiate data link layer connectivity. Each 802.11a/b/g device can operate in one of four possible modes:

1. ***Master mode*** (also called ***AP*** or ***infrastructure mode***) is used to create a service that looks like a traditional access point. The wireless card creates a network with a specified name (called the ***SSID***) and channel, and offers network services on it. While in master mode, wireless cards manage all communications related to the network (authenticating wireless clients, handling channel contention, repeating packets, etc.) Wireless cards in master mode can only communicate with cards that are associated with it in managed mode.
2. ***Managed mode*** is sometimes also referred to as ***client*** mode. Wireless cards in managed mode will join a network created by a master, and will automatically change their channel to match it. They then present any necessary credentials to the master, and if those credentials are accepted, they are said to be ***associated*** with the master. Managed mode cards do not communicate with each other directly, and will only communicate with an associated master.
3. ***Ad-hoc mode*** creates a multipoint-to-multipoint network where there is no single master node or AP. In ad-hoc mode, each wireless card communicates directly with its neighbors. Nodes must be in range of each other to communicate, and must agree on a network name and channel.
4. ***Monitor mode*** is used by some tools (such as Kismet, chapter six) to passively listen to all radio traffic on a given channel. When in monitor mode, wireless cards transmit no data. This is useful for analyzing problems on a wireless link or observing spectrum usage in the local area. Monitor mode is not used for normal communications.

When implementing a point-to-point or point-to-multipoint link, one radio will typically operate in master mode, while the other(s) operate in managed mode. In a multipoint-to-multipoint mesh, the radios all operate in ad-hoc mode so that they can communicate with each other directly.

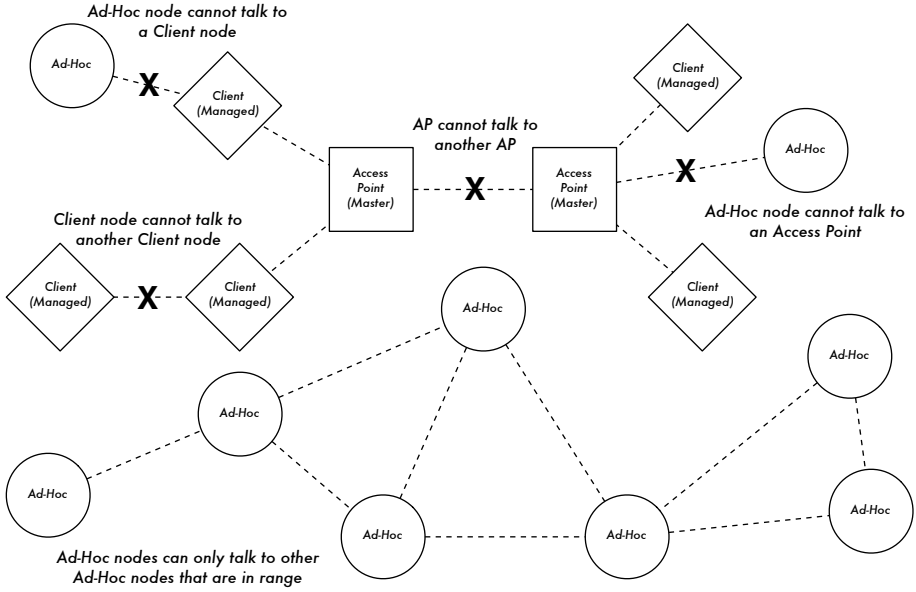

Figure 3.5: APs, Clients, and Ad-Hoc nodes.

It is important to keep these modes in mind when designing your network layout. Remember that managed mode clients cannot communicate with each other directly, so it is likely that you will want to run a high repeater site in master or ad-hoc mode. As we will see later in this chapter, ad-hoc is more flexible but has a number of performance issues as compared to using the master / managed modes.

Now that your wireless cards are providing physical and data link connectivity, they are ready to start passing around packets on layer 3: the internetworking layer.

Internet networking

IP addresses, network addressing, routing, and forwarding are important and related concepts in Internet networking. An ***IP address*** is an identifier for a network node such as a PC, server, router, or bridge. ***Network addressing*** is the system used to assign these identifiers in convenient groups. ***Routing*** keeps track of where in the network these groups may be found. The results of the routing process is kept in a list called a ***routing table***. ***Forwarding*** is the action of using the routing table to send a data packet to either the final destination or to the "next hop" which is closer to the destination.

IP addresses

In an IP[3] network, the address is a 32-bit number, normally written as four 8-bit numbers expressed in decimal form, separated by periods. Examples of IP addresses are 10.0.17.1, 192.168.1.1, or 172.16.5.23.

Network addressing

Interconnected networks must agree on an IP addressing plan. In the global Internet, committees of people allocate groups of IP addresses with a consistent, coherent method to ensure that duplicate addresses are not used by different networks and so that a shorthand can be used to refer to groups of addresses. These groups of addresses are called sub-networks, or ***subnets*** for short. Larger subnets can be further subdivided into smaller subnets. Sometimes a group of related addresses is referred to as an ***address space***.

On the Internet, no person or organization really owns these groups of addresses because the addresses only have meaning if the rest of the Internet community agrees with their usage. By agreement, the addresses are allocated to organizations according to their need and size. An organization which has been allocated an address range may then allocate a portion of that address range to another organization as part of a service agreement. Addresses which have been allocated in this manner, starting with internationally recognized committees, and then broken down hierarchically by national or smaller regional committees are referred to as ***globally routed IP addresses***.

Sometimes it is inconvenient or impossible to get more than one globally routed IP address allocated to an individual or organization. In this case a technique knows as Network Address Translation, or ***NAT*** can be used. A NAT device is a router with two network ports. The outside port uses one globally routed IP address, while the inside port uses an IP address from a special range known as ***private addresses***[4]. The NAT router allows the single global address to be shared with all of the inside users, who all use private addresses. It converts the packets from one form of addressing to the other as the packets pass through it. As far as the network users can tell, they are directly connected to the Internet and require no special software or drivers to share the single globally routed IP address.

---

3. In this book we deal primarily with IPv4, the version of the Internet Protocol in most common use today. While IPv6 will likely replace IPv4 at some point in the future, discussion of IPv6 is currently outside the scope of this book.

4. Private addresses are defined in RFC 1918, http://www.ietf.org/rfc/rfc1918

Routing

The Internet is constantly changing and growing. New networks are continually added, and links between networks are added and removed, fail and come back. It is the job of ***routing*** to determine the best path to the destination, and to create a routing table listing the best path for all the different destinations.

Static routing is the term used when the routing table is created by manual configuration. This is sometimes convenient for small networks but can easily become very difficult and error prone for large networks. Worse, if the best path to a network becomes unusable because of equipment failure or other reasons, static routing will not make use of the next best path.

Dynamic routing is a method in which network elements, in particular routers, exchange information about their state and the state of their neighbours in the network, and then use this information to automatically pick the best path and create the routing table. If something changes, such as a router failing or a new router being put into service, then the dynamic routing protocols make adjustments to the routing table. The system of packet exchanges and decision making is known as a ***routing protocol***. There are many routing protocols that are used in the Internet today, including OSPF, BGP, RIP, and EIGRP.

Wireless networks are like wired networks in that they need dynamic routing protocols, but they are also different enough from wired networks that they need different routing protocols. In particular, wired network connections typically work well or don't work at all (eg., either an Ethernet cable is plugged in, or it isn't). Things are not so clear when working with wireless networks. Wireless communication can be affected by objects moving into the path of the signal, or by interfering signals. Consequently, links may work well, or poorly, or vary between the two extremes. Since existing network protocols don't take the quality of a link into account when making routing decisions, the IEEE 802.11 committees and the IETF are working on standardizing protocols for wireless networks. It is currently unclear when a single standard for dealing with variable link quality will emerge.

In the meantime, there are many ongoing ad-hoc programming attempts to address the problem. Some examples include ***Hazy Sighted Link State*** (***HSLS***), ***Ad-hoc On-demand Distance Vector*** (***AODV***), and ***Optimized Link State Routing*** (***OLSR***). Another is ***SrcRR***, a combination of DSR and ETX implemented by the M.I.T. Roofnet project. Later in this chapter we will see an example of how to implement a network using OLSR to make routing decisions.

Forwarding

Forwarding is straightforward compared to addressing and routing. Each time a router receives a data packet, it consults its internal routing table. Starting with the high order (or most significant) bit, the routing table is searched for the entry that matches the most number of bits in the destination address. This is called the address ***prefix***. If an entry with a matching prefix is found in the routing table, then the ***hop count*** or ***time to live*** (***TTL***) field is decremented. If the result is zero, then the packet is dropped and an error packet is returned to the sender. Otherwise, the packet is sent to the node or interface specified in the routing table. For example, if the routing table contains these entries

```
Destination     Gateway         Genmask         Flags Metric  Iface
10.15.6.0       0.0.0.0         255.255.255.0   U     0       eth1
10.15.6.108     10.15.6.7       255.255.255.255 UG    1       eth1
216.231.38.0    0.0.0.0         255.255.255.0   U     0       eth0
0.0.0.0         216.231.38.1    0.0.0.0         UG    0       eth0
```

...and a packet arrives with the destination address of 10.15.6.23, then the router would send it out on interface eth1. If the packet has a destination of 10.15.6.108, then it would be forwarded to the gateway 10.15.6.7 (since it is more specific and matches more high-order bits than the 10.15.6.0 network route).

A destination of 0.0.0.0 is a special convention referred to as the ***default gateway***. If no other prefixes match the destination address, then the packet is sent to the default gateway. For example, if the destination address was 72.1.140.203, then the router would forward the packet to 216.231.38.1 (which would presumably send it closer to the ultimate destination, and so on).

If a packet arrives and no entry is found (i.e., there is no default gateway defined and no prefix matches a known route), then the packet is dropped and an error packet is returned to the sender.

The TTL field is used to detect routing loops. Without it, a packet could endlessly be sent back and forth between two routers who each list the other as the next best hop. These kinds of loops can cause so much unnecessary traffic on a network that they threaten its stability. Use of the TTL field doesn't fix routing loops, but it does help to prevent them from destroying a network due to simple misconfiguration.

Putting it all together

Once all network nodes have an IP address, they can send data packets to the IP address of any other node. Through the use of routing and forward-

ing, these packets can reach nodes on networks that are not physically connected to the originating node. This process describes much of what "happens" on the Internet. This is illustrated in the following figure:

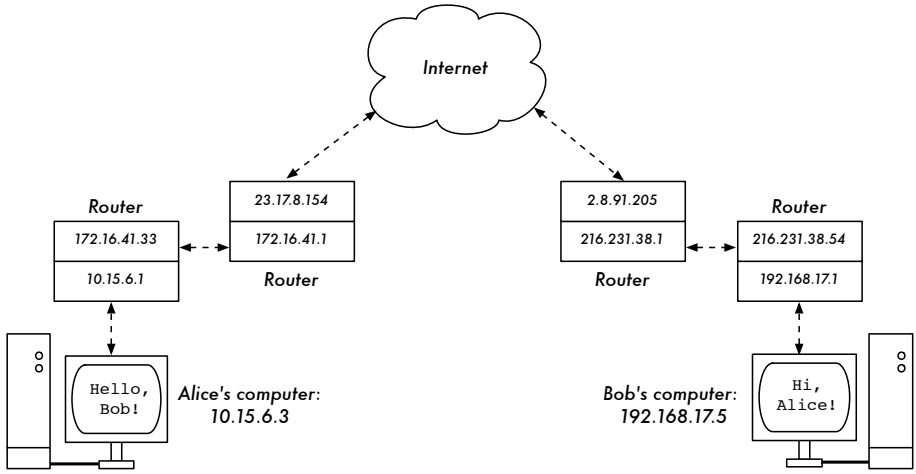

Figure 3.6: Internet networking. Each network segment has a router with two IP addresses, making it "link local" to two different networks. Packets are forwarded between routers until they reach their ultimate destination.

In this example, you can see the path that the packets take as Alice chats with Bob using an instant messaging service. Each dotted line represents an Ethernet cable, a wireless link, or any other kind of physical network. The cloud symbol is commonly used to stand in for "The Internet", and represents any number of intervening IP networks. Neither Alice nor Bob need to be concerned with how those networks operate, as long as the routers forward IP traffic towards the ultimate destination. If it weren't for Internet protocols and the cooperation of everyone on the net, this kind of communication would be impossible.

Now that we have seen how packets flow on IP networks, let's look at a very specialized kind of IP network: an OLSR mesh.

Mesh networking with OLSR

Most WiFi networks operate in infrastructure mode - they consist of an access point somewhere (with a radio operating in master mode), attached to a DSL line or other large scale wired network. In such a ***hotspot*** the access point usually acts as a master station that is distributing Internet access to its clients, which operate in managed mode. This topology is similar to a mobile phone (GSM) service. Mobile phones connect to a base station - without the presence of such a base station mobiles can't communicate with each other. If you make a joke call to a friend that is sitting on the other side of the table,

your phone sends data to the base station of your provider that may be a mile away - the base station then sends data back to the phone of your friend.

WiFi cards in managed mode can't communicate directly, either. Clients - for example, two laptops on the same table - have to use the access point as a relay. Any traffic between clients connected to an access point has to be sent twice. If client A and C communicate, client A sends data to the access point B, and then the access point will retransmit the data to client C. A single transmission may have a speed of 600 kByte/sec (thats about the maximum speed you could achieve with 802.11b) in our example - thus, because the data has to be repeated by the access point before it reaches its target, the effective speed between both clients will be only 300 kByte/sec.

In ad-hoc mode there is no hierarchical master-client relationship. Nodes can communicate directly as long as they are within the range of their wireless interfaces. Thus, in our example both computers could achieve full speed when operating ad-hoc, under ideal circumstances.

The disadvantage to ad-hoc mode is that clients do not repeat traffic destined for other clients. In the access point example, if two clients A and C can't directly “see” each other with their wireless interfaces, they still can communicate as long as the AP is in the wireless range of both clients.

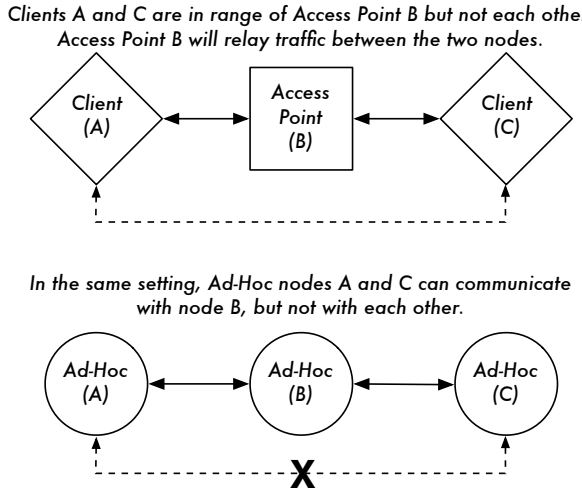

Figure 3.7: Access point B will relay traffic between clients A and C. In Ad-Hoc mode, node B will not relay traffic between A and C by default.

Ad-hoc nodes do not repeat by default, but they can effectively do the same if ***routing*** is applied. Mesh networks are based on the strategy that every mesh-enabled node acts as a relay to extend coverage of the wireless net-

work. The more nodes, the better the radio coverage and range of the mesh cloud.

There is one big tradeoff that must be mentioned at this point. If the device only uses one radio interface, the available bandwidth is significantly reduced every time traffic is repeated by intermediate nodes on the way from A to B. Also, there will be interference in transmission due to nodes sharing the same channel. Thus, cheap ad-hoc mesh networks can provide good radio coverage on the last mile(s) of a community wireless network at the cost of speed-- especially if the density of nodes and transmit power is high.

If an ad-hoc network consists of only a few nodes that are up and running at all time, don't move and always have stable radio links - a long list of ifs - it is possible to write individual routing tables for all nodes by hand.

Unfortunately, those conditions are rarely met in the real world. Nodes can fail, WiFi enabled devices roam around, and interference can make radio links unusable at any time. And no one wants to update several routing tables by hand if one node is added to the network. By using routing protocols that automatically maintain individual routing tables in all nodes involved, we can avoid these issues. Popular routing protocols from the wired world (such as OSPF) do not work well in such an environment because they are not designed to deal with lossy links or rapidly changing topology.

Mesh routing with olsrd

The Optimized Link State Routing Daemon - olsrd - from *olsr.org* is a routing application developed for routing in wireless networks. We will concentrate on this routing software for several reasons. It is a open-source project that supports Mac OS X, Windows 98, 2000, XP, Linux, FreeBSD, OpenBSD and NetBSD. Olsrd is available for access points that run Linux like the Linksys WRT54G, Asus Wl500g, AccessCube or Pocket PCs running Familiar Linux, and ships standard on Metrix kits running Metrix Pebble. Olsrd can handle multiple interfaces and is extensible with plug-ins. It supports IPv6 and it is actively developed and used by community networks all over the world.

Note that there are several implementations of Optimized Link State Routing, which began as an IETF-draft written at INRIA France. The implementation from *olsr.org* started as a master thesis of Andreas Toennesen at UniK University. Based on practical experience of the free networking community, the routing daemon was modified. Olsrd now differs significantly from the original draft because it includes a mechanism called Link Quality Extension that measures the packet loss between nodes and calculates routes according to this information. This extension breaks compatibility to routing daemons that follow the INRIA draft. The olsrd available from *olsr.org* can be configured to behave according to the IETF draft that lacks this feature - but there is no

reason to disable Link Quality Extension unless compliance with other implementations is required.

Theory

After olsrd is running for a while, a node knows about the existence of every other node in the mesh cloud and which nodes may be used to route traffic to them. Each node maintains a routing table covering the whole mesh cloud. This approach to mesh routing is called ***proactive routing***. In contrast, ***reactive routing*** algorithms seek routes only when it is necessary to send data to a specific node.

There are pros and cons to proactive routing, and there are many other ideas about how to do mesh routing that may be worth mentioning. The biggest advantage of proactive routing is that you know who is out there and you don't have to wait until a route is found. Higher protocol traffic overhead and more CPU load are among the disadvantages. In Berlin, the Freifunk community is operating a mesh cloud where olsrd has to manage more than 100 interfaces. The average CPU load caused by olsrd on a Linksys WRT54G running at 200 MHz is about 30% in the Berlin mesh. There is clearly a limit to what extent a proactive protocol can scale - depending on how many interfaces are involved and how often the routing tables are updated. Maintaining routes in a mesh cloud with static nodes takes less effort than a mesh with nodes that are constantly in motion, since the routing table has to be updated less often.

Mechanism

A node running olsrd is constantly broadcasting 'Hello' messages at a given interval so neighbours can detect it's presence. Every node computes a statistic how many 'Hellos' have been lost or received from each neighbour - thereby gaining information about the topology and link quality of nodes in the neighbourhood. The gained topology information is broadcasted as topology control messages (TC messages) and forwarded by neighbours that olsrd has chosen to be 'multipoint' relays.

The concept of multipoint relays is a new idea in proactive routing that came up with the OLSR draft. If every node rebroadcasts topology information that it has received, unnecessary overhead can be generated. Such transmissions are redundant if a node has many neighbours. Thus, an olsrd node decides which neighbours are favorable multipoint relays that should forward its topology control messages. Note that multipoint relays are only chosen for the purpose of forwarding TC messages. Payload is routed considering all available nodes.

Two other message types exist in OLSR that announce information: whether a node offers a gateway to other networks (HNA messages) or has multiple interfaces (MID messages). There is not much to say about what this messages do apart from the fact that they exist. HNA messages make olsrd very convenient when connecting to the Internet with a mobile device. When a mesh node roams around it will detect gateways into other networks and always choose the gateway that it has the best route to. However, olsrd is by no means bullet proof. If a node announces that it is an Internet gateway - which it isn't because it never was or it is just offline at the moment - the other nodes will nevertheless trust this information. The pseudo-gateway is a black hole. To overcome this problem, a dynamic gateway plugin was written. The plugin will automatically detect at the gateway if it is actually connected and whether the link is still up. If not, olsrd ceases to send false HNA messages. It is highly recommended to build and use this plugin instead of statically enabling HNA messages.

Practice

Olsrd implements IP-based routing in a userland application - installation is pretty easy. Installation packages are available for OpenWRT, AccessCube, Mac OS X, Debian GNU/Linux and Windows. OLSR is a standard part of Metrix Pebble. If you have to compile from source, please read the documentation that is shipped with the source package. If everything is configured properly all you have to do is start the olsr program.

First of all, it must be ensured that every node has a unique statically assigned IP-Address for each interface used for the mesh. It is not recommended (nor practicable) to use DHCP in an IP-based mesh network. A DHCP request will not be answered by a DHCP server if the node requesting DHCP needs a multihop link to connect to it, and applying dhcp relay throughout a mesh is likely impractical. This problem could be solved by using IPv6, since there is plenty of space available to generate a unique IP from the MAC address of each card involved (as suggested in "IPv6 Stateless Address Autoconfiguration in large mobile ad hoc networks" by K. Weniger and M. Zitterbart, 2002).

A wiki-page where every interested person can choose an individual IPv4 address for each interface the olsr daemon is running on may serve the purpose quite well. There is just not an easy way to automate the process if IPv4 is used.

The broadcast address should be 255.255.255.255 on mesh interfaces in general as a convention. There is no reason to enter the broadcast address explicitly, since olsrd can be configured to override the broadcast addresses with this default. It just has to be ensured that settings are the same everywhere. Olsrd can do this on its own. When a default olsrd configuration file is

issued, this feature should be enabled to avoid confusion of the kind 'why can't the other nodes see my machine?!?"

Now configure the wireless interface. Here is an example command how to configure a WiFi card with the name wlan0 using Linux:

```
iwconfig wlan0 essid olsr.org mode ad-hoc channel 10 rts 250 frag 256
```

Verify that the wireless part of the WiFi card has been configured so it has an ad-hoc connection to other mesh nodes within direct (single hop) range. Make sure the interface joins the same wireless channel, uses the same wireless network name ESSID (Extended Service Set IDentifier) and has the same Cell-ID as all other WiFi-Cards that build the mesh. Many WiFi cards or their respective drivers do not act compliant to the 802.11 standard for ad-hoc networking and thus may fail miserably to connect to a cell. They may be unable to connect to other devices on the same table, even if they are set up with the correct channel and wireless network name. They may rather confuse other cards that behave according to the standard by creating their own Cell-ID on the same channel with the same wireless network name. WiFi cards made by Intel that are shipped with Centrino Notebooks are notorious to do this.

You can check this out with the command **`iwconfig`** when using GNU-Linux. Here is the output on my machine:

```
wlan0   IEEE 802.11b  ESSID:"olsr.org"
        Mode:Ad-Hoc  Frequency:2.457 GHz  Cell: 02:00:81:1E:48:10
        Bit Rate:2 Mb/s   Sensitivity=1/3
        Retry min limit:8   RTS thr=250 B   Fragment thr=256 B
        Encryption key:off
        Power Management:off
        Link Quality=1/70  Signal level=-92 dBm  Noise level=-100 dBm
        Rx invalid nwid:0  Rx invalid crypt:28  Rx invalid frag:0
        Tx excessive retries:98024 Invalid misc:117503 Missed beacon:0
```

It is important to set the 'Request To Send' threshold value RTS for a mesh. There will be collisions on the radio channel between the transmissions of nodes on the same wireless channel, and RTS will mitigate this. RTS/CTS adds a handshake before each packet transmission to make sure that the channel is clear. This adds overhead, but increases performance in case of hidden nodes - and hidden nodes are the default in a mesh! This parameter sets the size of the smallest packet (in bytes) for which the node sends RTS. The RTS threshold value must be smaller than the IP-Packet size and the 'Fragmentation threshold' value - here set to 256 - otherwise it will be disabled. TCP is very sensitive to collisions, so it is important to switch RTS on.

Fragmentation allows to split an IP packet in a burst of smaller fragments transmitted on the medium. This adds overhead, but in a noisy environment

this reduces the error penalty and allows packets to get through interference bursts. Mesh networks are very noisy because nodes use the same channel and therefore transmissions are likely to interfere with each other. This parameter sets the maximum size before a data packet is split and sent in a burst - a value equal to the maximum IP packet size disables the mechanism, so it must be smaller than the IP packet size. Setting fragmentation threshold is recommended.

Once a valid IP-address and netmask is assigned and the wireless interface is up, the configuration file of olsrd must be altered in order that olsrd finds and uses the interfaces it is meant to work on.

For Mac OS-X and Windows there are nice GUI's for configuration and monitoring of the daemon available. Unfortunately this tempts users that lack background knowledge to do stupid things - like announcing black holes. On BSD and Linux the configuration file **`/etc/olsrd.conf`** has to be edited with a text editor.

A simple olsrd.conf

We are not going to provide a complete configuration file. Here are some essential settings that should be checked.

```
UseHysteresis           no
TcRedundancy            2
MprCoverage             3
LinkQualityLevel        2
LinkQualityWinSize      20

LoadPlugin "olsrd_dyn_gw.so.0.3"
{
        PlParam     "Interval"   "60"
        PlParam     "Ping"       "151.1.1.1"
        PlParam     "Ping"       "194.25.2.129"
}

Interface "ath0" "wlan0" {
        Ip4Broadcast 255.255.255.255
}
```

There are many more options available in the **`olsrd.conf`**, but these basic options should get you started. After these steps have been done, olsrd can be started with a simple command in a terminal:

```
olsrd -d 2
```

I recommend to run it with the debugging option -d 2 when used on a workstation, especially for the first time. You can see what olsrd does and monitor

how well the links to your neighbours are. On embedded devices the debug level should be 0 (off), because debugging creates a lot of CPU load.

The output should look something like this:

```
--- 19:27:45.51 ------------------------------------------- DIJKSTRA

192.168.120.1:1.00 (one-hop)
192.168.120.3:1.00 (one-hop)

--- 19:27:45.51 ---------------------------------------------- LINKS

IP address       hyst   LQ     lost   total  NLQ    ETX
192.168.120.1    0.000  1.000  0      20     1.000  1.00
192.168.120.3    0.000  1.000  0      20     1.000  1.00

--- 19:27:45.51 ------------------------------------------ NEIGHBORS

IP address       LQ     NLQ    SYM   MPR   MPRS  will
192.168.120.1    1.000  1.000  YES   NO    YES   3
192.168.120.3    1.000  1.000  YES   NO    YES   6

--- 19:27:45.51 ------------------------------------------- TOPOLOGY

Source IP addr   Dest IP addr     LQ     ILQ    ETX
192.168.120.1    192.168.120.17   1.000  1.000  1.00
192.168.120.3    192.168.120.17   1.000  1.000  1.00
```

Using OLSR on Ethernet and multiple interfaces

It is not necessary to have a wireless interface to test or use olsrd - although that is what olsrd is designed for. It may as well be used on any NIC. WiFi-interfaces don't have to operate always in ad-hoc mode to form a mesh when mesh nodes have more then one interface. For dedicated links it may be a very good option to have them running in infrastructure mode. Many WiFi cards and drivers are buggy in ad-hoc mode, but infrastructure mode works fine - because everybody expects at least this feature to work. Ad-hoc mode has not had many users so far, so the implementation of the ad-hoc mode was done sloppily by many manufacturers. With the rising popularity of mesh networks, the driver situation is improving now.

Many people use olsrd on wired and wireless interfaces - they don't think about network architecture. They just connect antennas to their WiFi cards, connect cables to their Ethernet cards, enable olsrd to run on all computers and all interfaces and fire it up. That is quite an abuse of a protocol that was designed to do wireless networking on lossy links - but - why not?

They expect olsrd to be an ueberprotocol. Clearly it is not necessary to send 'Hello' messages on a wired interface every two seconds - but it works. This

should not be taken as an recommendation - it is just amazing what people do with such a protocol and have success with it. In fact the idea of having a protocol that does everything for newbies that want to have a small to medium sized routed LAN is very appealing...

Plugins

A number of plugins are available for olsrd. Check out the *olsr.org* website for a complete list. Here a little HOWTO for the network topology visualization plugin **`olsrd_dot_draw`**.

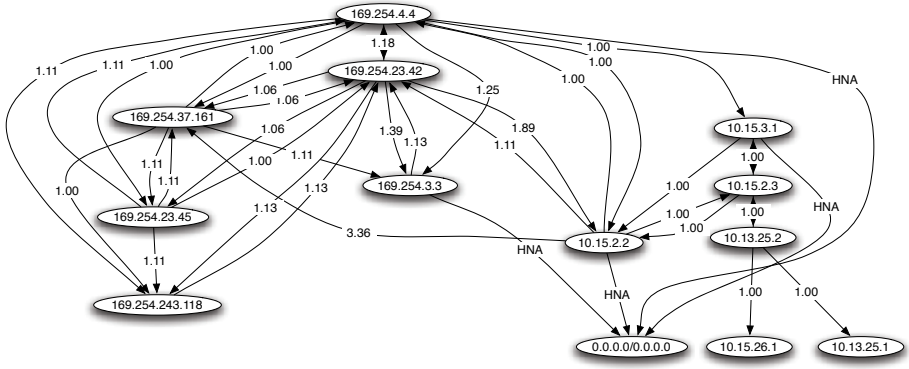

Figure 3.8: An automatically generated OLSR network topology.

Often it is very good for the understanding of a mesh network to have the ability to show the network topology graphically. Olsrd_dot_draw outputs the topology in the dot file format on TCP port 2004. The graphviz tools can then be used to draw the graphs.

Installing the dot_draw Plugin

Compile the olsr plugins separately and install them. To load the plugin add the following lines to **`/etc/olsrd.conf`**

```
LoadPlugin     "olsrd_dot_draw.so.0.3"
{
       PlParam "accept" "192.168.0.5"
       PlParam "port" "2004"
}
```

The parameter "accept" specifies which host is accepted to view the Topology Information (currently only one) and is "localhost" by default. The parameter "port" specifies the TCP port.

Then restart olsr and check if you get output on TCP Port 2004

```
telnet localhost 2004
```

After a while you should get some text output.

Now you can save the output graph descriptions and run the tools **`dot`** or **`neato`** form the graphviz package to get images.

Bruno Randolf has written a small perl script which continuously gets the topology information from olsrd and displays it using the graphviz and ImageMagick tools.

First install the following packages on your workstation:

- graphviz, *http://www.graphviz.org/*
- ImageMagick, *http://www.imagemagick.org/*

Download the script at: *http://meshcube.org/nylon/utils/olsr-topology-view.pl*

Now you can start the script with **`./olsr-topology-view.pl`** and view the topology updates in near-realtime.

Troubleshooting

As long as the WiFi-cards can 'see' each other directly with their radios, doing a ping will work whether olsrd is running or not. This works because the large netmasks effectively make every node link-local, so routing issues are sidestepped at the first hop. This should be checked first if things do not seem to work as expected. Most headaches people face with WiFi in Ad-Hoc mode are caused by the fact that the ad-hoc mode in drivers and cards are implemented sloppily. If it is not possible to ping nodes directly when they are in range it is most likely a card/driver issue, or your network settings are wrong.

If the machines can ping each other, but olsrd doesn't find routes, then the IP-addresses, netmask and broadcast address should be checked.

Are you running a firewall? Make sure it doesn't block UDP port 698.

Have fun!

Estimating capacity

Wireless links can provide significantly greater ***throughput*** to users than traditional Internet connections, such as VSAT, dialup, or DSL. Throughput is also referred to as ***channel capacity***, or simply ***bandwidth*** (although this term is unrelated to radio bandwidth). It is important to understand that a wireless device's listed speed (the ***data rate***) refers to the rate at which the radios can exchange symbols, not the usable throughput you will observe. As mentioned earlier, a single 802.11g link may use 54Mbps radios, but it will only provide up to 22Mbps of actual throughput. The rest is overhead that the radios need in order to coordinate their signals using the 802.11g protocol.

Note that throughput is a measurement of bits over time. 22Mbps means that in any given second, up to 22 megabits can be sent from one end of the link to the other. If users attempt to push more than 22 megabits through the link, it will take longer than one second. Since the data can't be sent immediately, it is put in a ***queue***, and transmitted as quickly as possible. This backlog of data increases the time needed for the most recently queued bits to the traverse the link. The time that it takes for data to traverse a link is called ***latency***, and high latency is commonly referred to as ***lag***. Your link will eventually send all of the queued traffic, but your users will likely complain as the lag increases.

How much throughput will your users really need? It depends on how many users you have, and how they use the wireless link. Various Internet applications require different amounts of throughput.

Application	BW / User	Notes
Text messaging / IM	< 1 Kbps	As traffic is infrequent and asynchronous, IM will tolerate high latency.
Email	1 to 100 Kbps	As with IM, email is asynchronous and intermittent, so it will tolerate latency. Large attachments, viruses, and spam significantly add to bandwidth usage. Note that web email services (such as Yahoo or Hotmail) should be considered as web browsing, not as email.

Application	BW / User	Notes
Web browsing	50 - 100+ Kbps	Web browsers only use the network when data is requested. Communication is asynchronous, so a fair amount of lag can be tolerated. As web browsers request more data (large images, long downloads, etc.) bandwidth usage will go up significantly.
Streaming audio	96 - 160 Kbps	Each user of a streaming audio service will use a constant amount of relatively large bandwidth for as long as it plays. It can tolerate some transient latency by using large buffers on the client. But extended periods of lag will cause audio "skips" or outright session failures.
Voice over IP (VoIP)	24 - 100+ Kbps	As with streaming audio, VoIP commits a constant amount of bandwidth to each user for the duration of the call. But with VoIP, the bandwidth is used roughly equally in both directions. Latency on a VoIP connection is immediate and annoying to users. Lag greater than a few milliseconds is unacceptable for VoIP.
Streaming video	64 - 200+ Kbps	As with streaming audio, some intermittent latency is avoided by using buffers on the client. Streaming video requires high throughput and low latency to work properly.
Peer-to-peer filesharing applications (BitTorrent, KaZaA, Gnutella, eDonkey, etc.)	0 - infinite Mbps	While peer to peer applications will tolerate any amount of latency, they tend to use up all available throughput by transmitting data to as many clients as possible, as quickly as possible. Use of these applications will cause latency and throughput problems for all other network users unless you use careful bandwidth shaping.

To estimate the necessary throughput you will need for your network, multiply the expected number of users by the sort of application they will likely use. For example, 50 users who are chiefly browsing the web will likely consume 2.5 to 5Mbps or more of throughput at peak times, and will tolerate some latency. On the other hand, 50 simultaneous VoIP users would require 5Mbps or more of throughput **in both directions** with absolutely no latency. Since 802.11g wireless equipment is ***half duplex*** (that is, it only transmits or receives, never both at once) you should accordingly double the required

throughput, for a total of **10Mbps**. Your wireless links must provide that capacity every second, or conversations will lag.

Since all of your users are unlikely to use the connection at precisely the same moment, it is common practice to ***oversubscribe*** available throughput by some factor (that is, allow more users than the maximum available bandwidth can support). Oversubscribing by a factor of 2 to 5 is quite common. In all likelihood, you will oversubscribe by some amount when building your network infrastructure. By carefully monitoring throughput throughout your network, you will be able to plan when to upgrade various parts of the network, and how much additional resources will be needed.

Expect that no matter how much capacity you supply, your users will eventually find applications that will use it all. As we'll see at the end of this chapter, using bandwidth shaping techniques can help mitigate some latency problems. By using bandwidth shaping, web caching, and other techniques, you can significantly reduce latency and increase overall network throughput.

To get a feeling for the lag felt on very slow connections, the ICTP has put together a bandwidth simulator. It will simultaneously download a web page at full speed and at a reduced rate that you choose. This demonstration gives you an immediate understanding of how low throughput and high latency reduce the usefulness of the Internet as a communications tool. It is available at *http://wireless.ictp.trieste.it/simulator/*

Link planning

A basic communication system consists of two radios, each with its associated antenna, the two being separated by the path to be covered. In order to have a communication between the two, the radios require a certain minimum signal to be collected by the antennas and presented to their input socket. Determining if the link is feasible is a process called ***link budget*** calculation. Whether or not signals can be passed between the radios depends on the quality of the equipment being used and on the diminishment of the signal due to distance, called ***path loss***.

Calculating the link budget

The power available in an 802.11 system can be characterized by the following factors:

- **Transmit Power**. It is expressed in milliwatts or in dBm. Transmit Power ranges from 30mW to 200mW or more. TX power is often dependent on the transmission rate. The TX power of a given device should be specified in the literature provided by the manufacturer, but can sometimes be diffi-

cult to find. Online databases such as the one provided by SeattleWireless (*http://www.seattlewireless.net/HardwareComparison*) may help.

- **Antenna Gain**. Antennas are passive devices that create the effect of amplification by virtue of their physical shape. Antennas have the same characteristics when receiving and transmitting. So a 12 dBi antenna is simply a 12 dBi antenna, without specifying if it is in transmission or reception mode. Parabolic antennas have a gain of 19-24 dBm, omnidirectional antennas have 5-12 dBi, sectorial antennas have roughly a 12-15 dBi gain.
- **Minimum Received Signal Level,** or simply, the sensitivity of the receiver. The minimum RSL is always expressed as a negative dBm (- dBm) and is the lowest power of signal the radio can distinguish. The minimum RSL is dependent upon rate, and as a general rule the lowest rate (1 Mbps) has the greatest sensitivity. The minimum will be typically in the range of -75 to -95 dBm. Like TX power, the RSL specifications should be provided by the manufacturer of the equipment.
- **Cable Losses**. Some of the signal's energy is lost in the cables, the connectors and other devices, going from the radios to the antennas. The loss depends on the type of cable used and on its length. Signal loss for short coaxial cables including connectors is quite low, in the range of 2-3 dB. It is better to have cables as short as possible.

When calculating the path loss, several effects must be considered. One has to take into account the ***free space loss***, ***attenuation*** and ***scattering***. Signal power is diminished by geometric spreading of the wavefront, commonly known as free space loss. Ignoring everything else, the further away the two radios, the smaller the received signal is due to free space loss. This is independent from the environment, depending only on the distance. This loss happens because the radiated signal energy expands as a function of the distance from the transmitter.

Using decibels to express the loss and using 2.45 GHz as the signal frequency, the equation for the free space loss is

```
Lfsl = 40 + 20*log(r)
```

where Lfsl is expressed in dB and r is the distance between the transmitter and receiver, in meters.

The second contribution to the path loss is given by attenuation. This takes place as some of the signal power is absorbed when the wave passes through solid objects such as trees, walls, windows and floors of buildings. Attenuation can vary greatly depending upon the structure of the object the signal is passing through, and it is very difficult to quantify. The most convenient way to express its contribution to the total loss is by adding an "allowed loss" to the free space. For example, experience shows that trees add

10 to 20 dB of loss per tree in the direct path, while walls contribute 10 to 15 dB depending upon the construction.

Along the link path, the RF energy leaves the transmitting antenna and energy spreads out. Some of the RF energy reaches the receiving antenna directly, while some bounces off the ground. Part of the RF energy which bounces off the ground reaches the receiving antenna. Since the reflected signal has a longer way to travel, it arrives at the receiving antenna later than the direct signal. This effect is called ***multipath***, fading or signal dispersion. In some cases reflected signals add together and cause no problem. When they add together out of phase, the received signal is almost worthless. In same cases, the signal at the receiving antenna can be zeroed by the reflected signals. This is known as ***nulling***. There is a simple technique that is used to deal with multipath, called ***antenna diversity***. It consists in adding a second antenna to the radio. Multipath is in fact a very location-specific phenomenon. If two signals add out of phase at one location, they will not add destructively at a second, nearby location. If there are two antennas, at least one of them should be able to receive a usable signal, even if the other is receiving a distorted one. In commercial devices, antenna switching diversity is used: there are multiple antennas on multiple inputs, with a single receiver. The signal is thus received through only one antenna at a time. When transmitting, the radio uses the antenna last used for reception. The distortion given by multipath degrades the ability of the receiver to recover the signal in a manner much like signal loss. A simple way of applying the effects of scattering in the calculation of the path loss is to change the exponent of the distance factor of the free space loss formula. The exponent tends to increase with the range in an environment with a lot of scattering. An exponent of 3 can be used in an outdoor environment with trees, while one of 4 can be used for an indoor environment.

When free space loss, attenuation, and scattering are combined, the path loss is:

```
L(dB) = 40 + 10*n*log(r) + L(allowed)
```

For a rough estimate of the link feasibility, one can evaluate just the free space loss. The environment can bring further signal loss, and should be considered for an exact evaluation of the link. The environment is in fact a very important factor, and should never be neglected.

To evaluate if a link is feasible, one must know the characteristics of the equipment being used and evaluate the path loss. Note that when performing this calculation, you should only add the TX power of one side of the link. If you are using different radios on either side of the link, you should calculate the path loss twice, once for each direction (using the appropriate TX power

for each calculation). Adding up all the gains and subtracting all the losses gives

```
  TX Power Radio 1
+ Antenna Gain Radio 1
- Cable Losses Radio 1
+ Antenna Gain Radio 2
- Cable Losses Radio 2
_____________________________

= Total Gain
```

Subtracting the Path Loss from the Total Gain:

```
  Total Gain
- Path Loss
_____________

= Signal Level at one side of the link
```

If the resulting signal level is greater than the minimum received signal level, then the link is feasible! The received signal is powerful enough for the radios to use it. Remember that the minimum RSL is always expressed as a negative dBm, so -56dBm is greater than -70dBm. On a given path, the variation in path loss over a period of time can be large, so a certain margin (difference between the signal level and the minimum received signal level) should be considered. This margin is the amount of signal above the sensitivity of radio that should be received in order to ensure a stable, high quality radio link during bad weather and other atmospheric disturbances. A margin of error of 10-15 dB is fine. To give some space for attenuation and multipath in the received radio signal, a margin of 20dB should be safe enough.

Once you have calculated the link budget in one direction, repeat the calculation for the other direction. Substitute the transmit power for that of the second radio, and compare the result against the minimum received signal level of the first radio.

Example link budget calculation

As an example, we want to estimate the feasibility of a 5km link, with one access point and one client radio. The access point is connected to an omnidirectional antenna with 10dBi gain, while the client is connected to a sectorial antenna with 14dBi gain. The transmitting power of the AP is 100mW (or 20dBm) and its sensitivity is -89dBm. The transmitting power of the client is 30mW (or 15dBm) and its sensitivity is -82dBm. The cables are short, with a loss of 2dB at each side.

Adding up all the gains and subtracting all the losses for the AP to client link gives:

```
  20 dBm (TX Power Radio 1)
+ 10 dBi (Antenna Gain Radio 1)
-  2 dB  (Cable Losses Radio 1)
+ 14 dBi (Antenna Gain Radio 2)
-  2 dB  (Cable Losses Radio 2)
————————
  40 dB = Total Gain
```

The path loss for a 5km link, considering only the free space loss is:

```
Path Loss = 40 + 20log(5000) = 113 dB
```

Subtracting the path loss from the total gain

```
40 dB - 113 dB = -73 dB
```

Since -73dB is greater than the minimum receive sensitivity of the client radio (-82dBm), the signal level is just enough for the client radio to be able to hear the access point. There is only 9dB of margin (82dB - 73dB) which will likely work fine in fair weather, but may not be enough to protect against extreme weather conditions.

Next we calculate the link from the client back to the access point:

```
  15 dBm (TX Power Radio 2)
+ 14 dBi (Antenna Gain Radio 2)
-  2 dB  (Cable Losses Radio 2)
+ 10 dBi (Antenna Gain Radio 1)
-  2 dB  (Cable Losses Radio 1)
————————
  35 dB = Total Gain
```

Obviously, the path loss is the same on the return trip. So our received signal level on the access point side is:

```
35 dB - 113 dB = -78 dB
```

Since the receive sensitivity of the AP is -89dBm, this leaves us 11dB of fade margin (89dB - 78dB). Overall, this link will probably work but could use a bit more gain. By using a 24dBi dish on the client side rather than a 14dBi sectorial antenna, you will get an additional 10dBi of gain on both directions of the link (remember, antenna gain is reciprocal). A more expensive option would be to use higher power radios on both ends of the link, but note that adding an amplifier or higher powered card to one end does not help the overall quality of the link.

Online tools can be used to calculate the link budget. For example, the Green Bay Professional Packet Radio's Wireless Network Link Analysis (*http://my.athenet.net/~multiplx/cgi-bin/wireless.main.cgi*) is an excellent tool. The Super Edition generates a PDF file containing the Fresnel zone and radio path graphs. The calculation scripts can even be downloaded from the website and installed locally. We will look at one excellent online tool in more detail in the next section, **Link planning software**.

The Terabeam website also has excellent calculators available online (*http://www.terabeam.com/support/calculations/index.php*).

Tables for calculating link budget

To calculate the link budget, simply approximate your link distance, then fill in the following tables:

Free Space Path Loss at 2.4GHz

Distance (m)	100	500	1,000	3,000	5,000	10,000
Loss (dB)	80	94	100	110	113	120

Antenna Gain:

Radio 1 Antenna (dBi)	+ Radio 2 Antenna (dBi)	= Total Antenna Gain

Losses:

Radio 1 + Cable Loss (dB)	Radio 2 + Cable Loss (dB)	Free Space Path Loss (dB)	= Total Loss (dB)

Link Budget for Radio 1 ➞ Radio 2:

Radio 1 TX Power	+ Antenna Gain	- Total Loss	= Signal	> Radio 2 Sensitivity

Link Budget for Radio 2 ➞ Radio 1:

Radio 2 TX Power	+ Antenna Gain	- Total Loss	= Signal	> Radio 1 Sensitivity

If the received signal is greater than the minimum received signal strength in both directions of the link, then the link is feasible.

Link planning software

While calculating a link budget by hand is straightforward, there are a number of tools available that will help automate the process. In addition to calculating free space loss, these tools will take many other relevant factors into account as well (such as tree absorption, terrain effects, climate, and even estimating path loss in urban areas). In this section, we will discuss two free tools that are useful for planning wireless links: Green Bay Professional Packet Radio's online interactive network design utilities, and RadioMobile.

Interactive design CGIs

The Green Bay Professional Packet Radio group (GBPRR) has made a variety of very useful link planning tools available for free online. You can browse these tools online at *http://www.qsl.net/n9zia/wireless/page09.html* . Since the tools are available online, they will work with any device that has a web browser and Internet access.

We will look at the first tool, **Wireless Network Link Analysis**, in detail. You can find it online at *http://my.athenet.net/~multiplx/cgi-bin/wireless.main.cgi* .

To begin, enter the channel to be used on the link. This can be specified in MHz or GHz. If you don't know the frequency, consult the table in Appendix B. Note that the table lists the channel's center frequency, while the tool asks for the highest transmitted frequency. The difference in the ultimate

result is minimal, so feel free to use the center frequency instead. To find the highest transmitted frequency for a channel, just add 11MHz to the center frequency.

Next, enter the details for the transmitter side of the link, including the transmission line type, antenna gain, and other details. Try to fill in as much data as you know or can estimate. You can also enter the antenna height and elevation for this site. This data will be used for calculating the antenna tilt angle. For calculating Fresnel zone clearance, you will need to use GBPRR's Fresnel Zone Calculator.

The next section is very similar, but includes information about the other end of the link. Enter all available data in the appropriate fields.

Finally, the last section describes the climate, terrain, and distance of the link. Enter as much data as you know or can estimate. Link distance can be calculated by specifying the latitude and longitude of both sites, or entered by hand.

Now, click the Submit button for a detailed report about the proposed link. This includes all of the data entered, as well as the projected path loss, error rates, and uptime. These numbers are all completely theoretical, but will give you a rough idea of the feasibility of the link. By adjusting values on the form, you can play "what-if?" to see how changing various parameters will affect the connection.

In addition to the basic link analysis tool, GBPRR provides a "super edition" that will produce a PDF report, as well as a number of other very useful tools (including the Fresnel Zone Calculator, Distance & Bearing Calculator, and Decibel Conversion Calculator to name just a few). Source code to most of the tools is provided as well.

RadioMobile

Radio Mobile is a tool for the design and simulation of wireless systems. It predicts the performance of a radio link by using information about the equipment and a digital map of the area. It is public domain software that runs on Windows, or using Linux and the Wine emulator.

Radio Mobile uses a ***digital terrain elevation model*** for the calculation of coverage, indicating received signal strength at various points along the path. It automatically builds a profile between two points in the digital map showing the coverage area and first Fresnel zone. During the simulation, it checks for line of sight and calculates the Path Loss, including losses due to obstacles. It is possible to create networks of different topologies, including net master/slave, point-to-point, and point-to-multipoint.

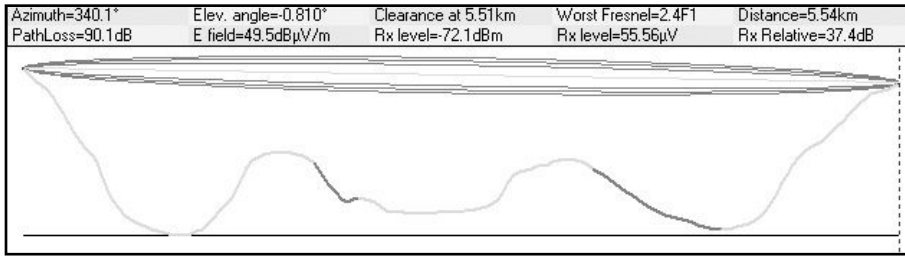

Figure 3.9: Link feasibility, including Fresnel zone and line of sight estimate, using RadioMobile.

The software calculates the coverage area from the base station in a point-to-multipoint system. It works for systems having frequencies from 20 kHz to 200 GHz. ***Digital elevation maps*** (***DEM***) are available for free from several sources, and are available for most of the world. DEMs do not show coastlines or other readily identifiable landmarks, but they can easily be combined with other kinds of data (such as aerial photos or topographical charts) in several layers to obtain a more useful and readily recognizable representation. You can digitize your own maps and combine them with DEMs. The digital elevation maps can be merged with scanned maps, satellite photos and Internet map services (such as Mapquest) to produce accurate prediction plots.

Download Radio Mobile here: *http://www.cplus.org/rmw/download.html*

The main Radio Mobile webpage, with examples and tutorials, is available at: *http://www.cplus.org/rmw/english1.html*

RadioMobile under Linux

Radio Mobile will also work using Wine under Ubuntu Linux. While the application runs, some button labels may run beyond the frame of the button and can be hard to read.

We were able to make Radio Mobile work with Linux using the following environment:

- IBM Thinkpad x31
- Ubuntu Breezy (v5.10), *http://www.ubuntu.com/*
- Wine version 20050725, from the Ubuntu Universe repository

There are detailed instructions for installing RadioMobile on Windows at *http://www.cplus.org/rmw/download.html*. You should follow all of the steps except for step 1 (since it is difficult to extract a DLL from the VBRUN60SP6.EXE file under Linux). You will either need to copy the MSVBVM60.DLL file from a Windows machine that already has the Visual

Basic 6 run-time environment installed, or simply Google for MSVBVM60.DLL, and download the file.

Now continue with step 2 at from the above URL, making sure to unzip the downloaded files in the same directory into which you have placed the downloaded DLL file. Note that you don't have to worry about the stuff after step 4; these are extra steps only needed for Windows users.

Finally, you can start Wine from a terminal with the command:

```
# wine RMWDLX.exe
```

You should see RadioMobile running happily in your XWindows session.

Avoiding noise

The unlicensed ISM and U-NII bands represent a very tiny piece of the known electromagnetic spectrum. Since this region can be utilized without paying license fees, many consumer devices use it for a wide range of applications. Cordless phones, analog video senders, Bluetooth, baby monitors, and even microwave ovens compete with wireless data networks for use of the very limited 2.4GHz band. These signals, as well as other local wireless networks, can cause significant problems for long range wireless links. Here are some steps you can use to reduce reception of unwanted signals.

- **Increase antenna gain on both sides of a point-to-point link.** Antennas not only add gain to a link, but their increased directionality tends to reject noise from areas around the link. Two high gain dishes that are pointed at each other will reject noise from directions that are outside the path of the link. Using omnidirectional antennas will receive noise from all directions.
- **Don't use an amplifier.** As we will see in chapter four, amplifiers can make interference issues worse by indiscriminately amplifying all received signals, including sources of interference. Amplifiers also cause interference problems for other nearby users of the band.
- **Use sectorials instead of using an omnidirectional.** By making use of several sectorial antennas, you can reduce the overall noise received at a distribution point. By staggering the channels used on each sectorial, you can also increase the available bandwidth to your clients.

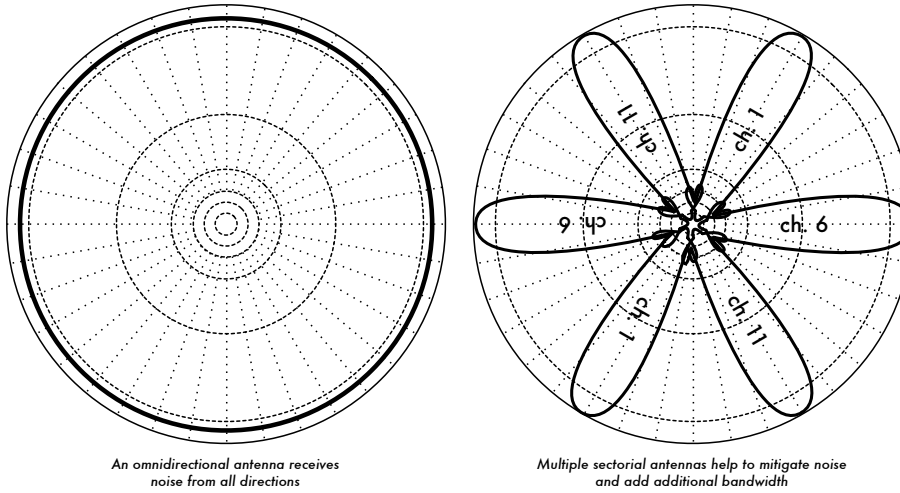

Figure 3.10: A single omnidirectional antenna vs. multiple sectorials.

- **Use the best available channel.** Remember that 802.11b/g channels are 22Mhz wide, but are only separated by 5MHz. Perform a site survey (as detailed in chapter eight), and select a channel that is as far as possible from existing sources of interference. Remember that the wireless landscape can change at any time as people add new devices (cordless phones, other networks, etc.) If your link suddenly has trouble sending packets, you may need to perform another site survey and pick a different channel.
- **Use smaller hops and repeaters, rather than a single long distance shot.** Keep your point-to-point links as short as possible. While it may be possible to create a 12km link that cuts across the middle of a city, you will likely have all kinds of interference problems. If you can break that link into two or three shorter hops, the link will likely be more stable. Obviously this isn't possible on long distance rural links where power and mounting structures are unavailable, but noise problems are also unlikely in those settings.
- **If possible, use 5.8GHz, 900MHz, or another unlicensed band**. While this is only a short term solution, there is currently far more consumer equipment installed in the field that uses 2.4GHz. Using 802.11a or a 2.4GHz to 5.8GHz step-up device will let you avoid this congestion altogether. If you can find it, some old 802.11 equipment uses unlicensed spectrum at 900MHz (unfortunately at much lower bit rates). Other technologies, such as Ronja (*http://ronja.twibright.com/*) use optical technology for short distance, noise-free links.
- **If all else fails, use licensed spectrum**. There are places where all available unlicensed spectrum is effectively used. In these cases, it may make sense to spend the additional money for proprietary equipment that

uses a less congested band. For long distance point-to-point links that require very high throughput and maximum uptime, this is certainly an option. Of course, these features come at a much higher price tag compared to unlicensed equipment.

To identify sources of noise, you need tools that will show you what is happening in the air at 2.4GHz. We will see some examples of these tools in chapter six.

Repeaters

The most critical component to building long distance network links is ***line of sight*** (often abbreviated as ***LOS***). Terrestrial microwave systems simply cannot tolerate large hills, trees, or other obstacles in the path of a long distance link. You must have a clear idea of the lay of the land between two points before you can determine if a link is even possible.

But even if there is a mountain between two points, remember that obstacles can sometimes be turned into assets. Mountains may block your signal, but assuming power can be provided they also make very good ***repeater*** sites.

Repeaters are nodes that are configured to rebroadcast traffic that is not destined for the node itself. In a mesh network, every node is a repeater. In a traditional infrastructure network, nodes must be configured to pass along traffic to other nodes.

A repeater can use one or more wireless devices. When using a single radio (called a ***one-arm repeater***), overall efficiency is slightly less than half of the available bandwidth, since the radio can either send or receive data, but never both at once. These devices are cheaper, simpler, and have lower power requirements. A repeater with two (or more) radio cards can operate all radios at full capacity, as long as they are each configured to use non-overlapping channels. Of course, repeaters can also supply an Ethernet connection to provide local connectivity.

Repeaters can be purchased as a complete hardware solution, or easily assembled by connecting two or more wireless nodes together with Ethernet cable. When planning to use a repeater built with 802.11 technology, remember that nodes must be configured for master, managed, or ad-hoc mode. Typically, both radios in a repeater are configured for master mode, to allow multiple clients to connect to either side of the repeater. But depending on your network layout, one or more devices may need to use ad-hoc or even client mode.

Typically, repeaters are used to overcome obstacles in the path of a long distance link. For example, there may be buildings in your path, but those

buildings contain people. Arrangements can often be worked out with building owners to provide bandwidth in exchange for roof rights and electricity. If the building owner isn't interested, tenants on high floors may be able to be persuaded to install equipment in a window.

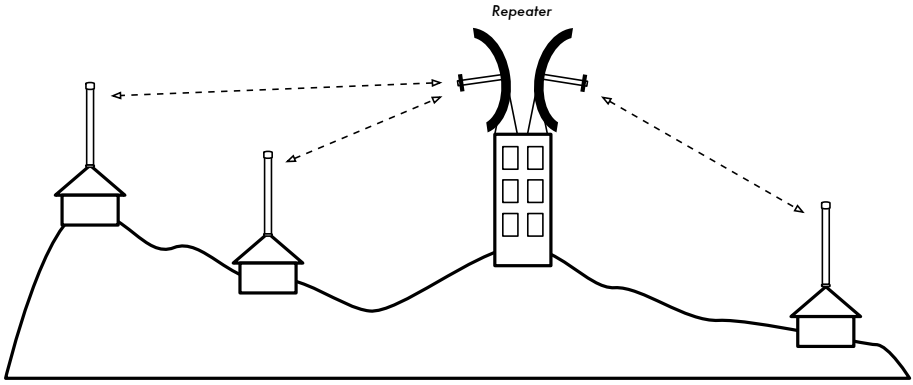

Figure 3.11: The repeater forwards packets over the air between nodes that have no direct line of sight.

If you can't go over or through an obstacle, you can often go around it. Rather than using a direct link, try a multi-hop approach to avoid the obstacle.

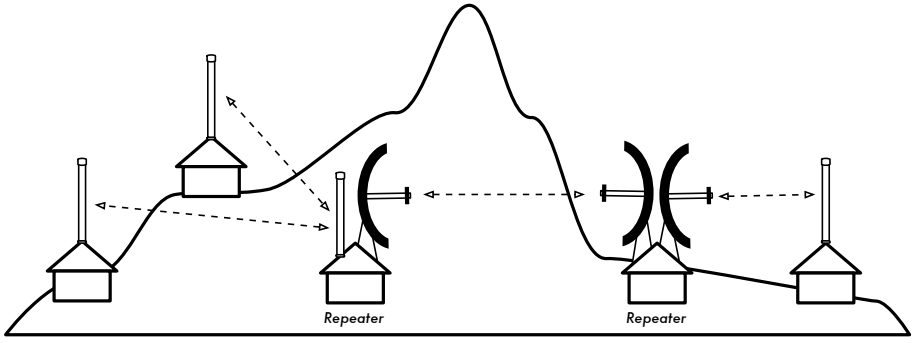

Figure 3.12: No power was available at the top of the hill, but it was circumvented by using multiple repeater sites around the base.

Finally, you may need to consider going backwards in order to go forwards. If there is a high site available in a different direction, and that site can see beyond the obstacle, a stable link can be made via an indirect route.

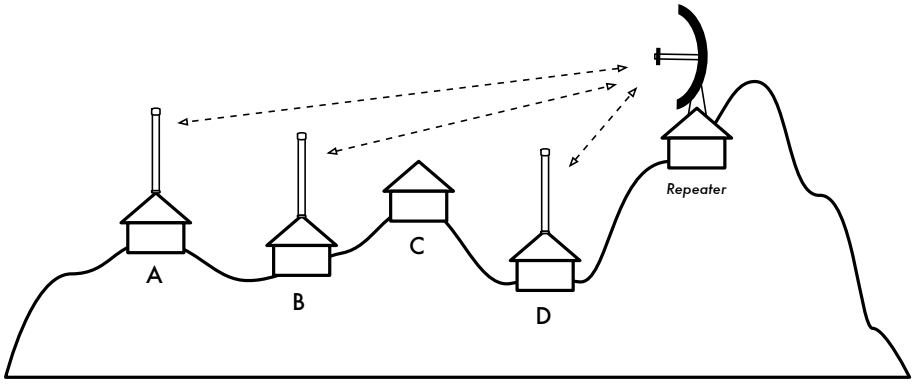

Figure 3.13: Site D could not make a clean link to site A or B, since site C is in the way and is not interested in hosting a node. By installing a high repeater, nodes A, B, and D can communicate. Note that traffic from node D actually travels further away from the rest of the network before the repeater forwards it along.

Repeaters in networks remind me of the "six degrees of separation" principle. This idea says that no matter who you are looking for, you need only contact five intermediaries before finding the person. Repeaters in high places can "see" a great deal of intermediaries, and as long as your node is in range of the repeater, you can communicate with any node the repeater can reach.

Traffic optimization

Bandwidth is measured as a bit rate over a time interval. This means that over time, bandwidth available on any link approaches infinity. Unfortunately, for any given period of time, the bandwidth provided by any given network connection is not infinite. You can always download (or upload) as much traffic as you like; you need only wait long enough. Of course, human users are not as patient as computers, and are not willing to wait an infinite amount of time for their information to traverse the network. For this reason, bandwidth must be managed and prioritized much like any other limited resource.

You will significantly improve response time and maximize available throughput by eliminating unwanted and redundant traffic from your network. This section describes many common techniques for making sure that your network carries only the traffic that must traverse it.

Web caching

A web proxy server is a server on the local network that keeps copies of recently retrieved or often used web pages, or parts of pages. When the next person retrieves these pages, they are served from the local proxy server instead of from the Internet. This results in significantly faster web access in most cases, while reducing overall Internet bandwidth usage. When a proxy

server is implemented, the administrator should also be aware that some pages are not cacheable-- for example, pages that are the output of server-side scripts, or other dynamically generated content.

The apparent loading of web pages is also affected. With a slow Internet link, a typical page begins to load slowly, first showing some text and then displaying the graphics one by one. In a network with a proxy server, there could be a delay when nothing seems to happen, and then the page will load almost at once. This happens because the information is sent to the computer so quickly that it spends a perceptible amount of time rendering the page. The overall time it takes to load the whole page might take only ten seconds (whereas without a proxy server, it may take 30 seconds to load the page gradually). But unless this is explained to some impatient users, they may say the proxy server has made things slower. It is usually the task of the network administrator to deal with user perception issues like these.

Proxy server products

There are a number of web proxy servers available. These are the most commonly used software packages:

- **Squid**. Open source Squid is the de facto standard at universities. It is free, reliable, easy to use and can be enhanced (for example, adding content filtering and advertisement blocking). Squid produces logs that can be analyzed using software such as Awstats, or Webalizer, both of which are open source and produce good graphical reports. In most cases, it is easier to install as part of the distribution than to download it from *http://www.squid-cache.org/* (most Linux distributions such as Debian, as well as other versions of Unix such as NetBSD and FreeBSD come with Squid). A good Squid configuration guide can be found at *http://squid-docs.sourceforge.net/latest/book-full.html*.
- **Microsoft Proxy server 2.0.** Not available for new installations because it has been superseded by Microsoft ISA server and is no longer supported. It is nonetheless used by some institutions, although it should perhaps not be considered for new installations.
- **Microsoft ISA server.** ISA server is a very good proxy server program, that is arguably too expensive for what it does. However, with academic discounts it may be affordable to some institutions. It produces its own graphical reports, but its log files can also be analyzed with popular analyzer software such as Sawmill (*http://www.sawmill.net/*). Administrators at a site with MS ISA Server should spend sufficient time getting the configuration right; otherwise MS ISA Server can itself be a considerable bandwidth user. For example, a default installation can easily consume more bandwidth than the site has used before, because popular pages with short expiry dates (such as news sites) are continually being refreshed. There-

fore it is important to get the pre-fetching settings right, and to configure pre-fetching to take place mainly overnight. ISA Server can also be tied to content filtering products such as WebSense. For more information, see: *http://www.microsoft.com/isaserver/* and *http://www.isaserver.org/* .

Preventing users from bypassing the proxy server

While circumventing Internet censorship and restrictive information access policy may be a laudable political effort, proxies and firewalls are necessary tools in areas with extremely limited bandwidth. Without them, the stability and usability of the network are threatened by legitimate users themselves. Techniques for bypassing a proxy server can be found at *http://www.antiproxy.com/* . This site is useful for administrators to see how their network measures up against these techniques.

To enforce use of the caching proxy, you might consider simply setting up a network access policy and trusting your users. In the layout below, the administrator has to trust that his users will not bypass the proxy server.

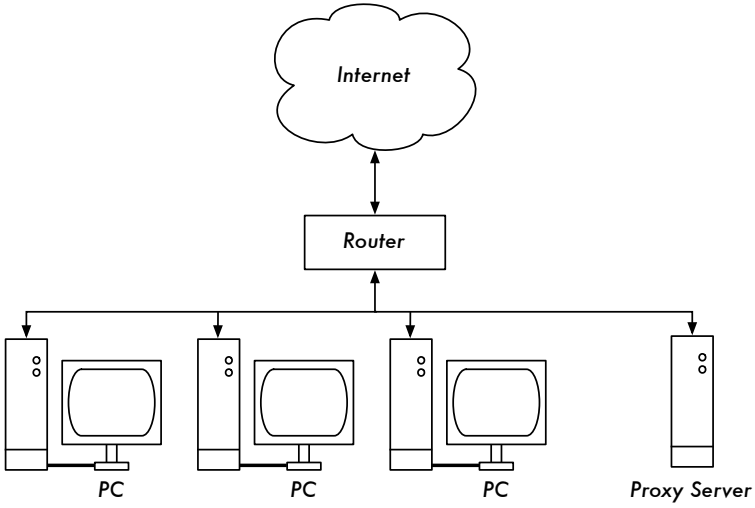

Figure 3.14: This network relies on trusted users to properly configure their PCs to use the proxy server.

In this case the administrator typically uses one of the following techniques:

- **Not giving out the default gateway address through DCHP.** This may work for a while, but some network-savvy users who want to bypass the proxy might find or guess the default gateway address. Once that happens, word tends to spread about how to bypass the proxy.
- **Using domain or group policies.** This is very useful for configuring the correct proxy server settings for Internet Explorer on all computers in the

domain, but is not very useful for preventing the proxy from being bypassed, because it depends on a user logging on to the NT domain. A user with a Windows 95/98/ME computer can cancel his log-on and then bypass the proxy, and someone who knows a local user password on his Windows NT/2000/XP computer can log on locally and do the same.

- **Begging and fighting with users.** This is never an optimal situation for a network administrator.

The only way to ensure that proxies cannot be bypassed is by using the correct network layout, by using one of the three techniques described below.

Firewall

A more reliable way to ensure that PCs don't bypass the proxy can be implemented using the firewall. The firewall can be configured to allow only the proxy server through, i.e. to make HTTP requests to the Internet. All other PCs are blocked, as shown in the diagram below.

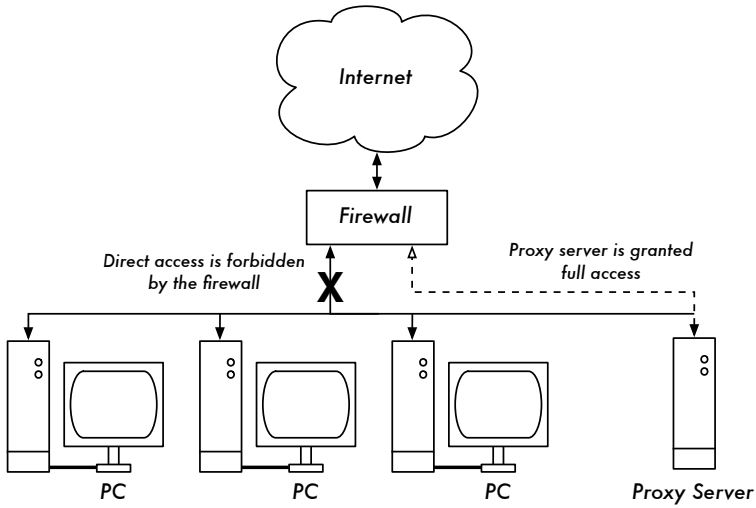

Figure 3.15: The firewall prevents PCs from accessing the Internet directly, but allows access via the proxy server.

Relying on a firewall, as in the above diagram, may or may not be sufficient, depending on how the firewall is configured. If it only blocks access from the campus LAN to port 80 on web servers, there will be ways for clever users to find ways around it. Additionally, they will be able to use other bandwidth hungry protocols such as Kazaa.

Two network cards

Perhaps the most reliable method is to install two network cards in the proxy server and connect the campus network to the Internet as shown below. In this way, the network layout makes it physically impossible to reach the Internet without going through the proxy server.

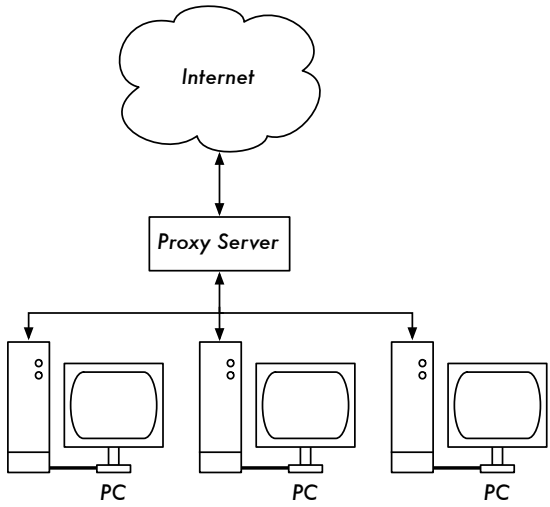

Figure 3.16: The only route to the Internet is through the proxy.

The proxy server in this diagram should not have IP forwarding enabled, unless the administrators knows exactly what they want to let through.

One big advantage to this design is that a technique known as ***transparent proxying*** can be used. Using a transparent proxy means that users' web requests are automatically forwarded to the proxy server, without any need to manually configure web browsers to use it. This effectively forces all web traffic to be cached, eliminates many chances for user error, and will even work with devices that do not support use of a manual proxy. For more details about configuring a transparent proxy with Squid, see:

- *http://www.squid-cache.org/Doc/FAQ/FAQ-17.html*
- *http://en.tldp.org/HOWTO/mini/TransparentProxy-2.html*

Policy-based routing

One way to prevent bypassing of the proxy using Cisco equipment is with policy routing. The Cisco router transparently directs web requests to the proxy server. This technique is used at Makerere University. The advantage

of this method is that, if the proxy server is down, the policy routes can be temporarily removed, allowing clients to connect directly to the Internet.

Mirroring a website

With permission of the owner or web master of a site, the whole site can be mirrored to a local server overnight, if it is not too large. This is something that might be considered for important websites that are of particular interest to the organization or that are very popular with web users. This may have some use, but it has some potential pitfalls. For example, if the site that is mirrored contains CGI scripts or other dynamic content that require interactive input from the user, this would cause problems. An example is a website that requires people to register online for a conference. If someone registers online on a mirrored server (and the mirrored script works), the organizers of the site will not have the information that the person registered.

Because mirroring a site may infringe copyright, this technique should only be used with permission of the site concerned. If the site runs ***rsync***, the site could be mirrored using rsync. This is likely the fastest and most efficient way to keep site contents synchronized. If the remote web server is not running rsync, the recommended software to use is a program called ***wget***. It is part of most versions of Unix/Linux. A Windows version can be found at *http://xoomer.virgilio.it/hherold/*, or in the free Cygwin Unix tools package (*http://www.cygwin.com/*).

A script can be set up to run every night on a local web server and do the following:

- Change directory to the web server document root: for example, **`/var/www/`** on Unix, or **`C:\Inetpub\wwwroot`** on Windows.
- Mirror the website using the command:

```
wget --cache=off -m http://www.python.org
```

The mirrored website will be in a directory **`www.python.org`**. The web server should now be configured to serve the contents of that directory as a name-based virtual host. Set up the local DNS server to fake an entry for this site. For this to work, client PCs should be configured to use the local DNS server(s) as the primary DNS. (This is advisable in any case, because a local caching DNS server speeds up web response times).

Pre-populate the cache using wget

Instead of setting up a mirrored website as described in the previous section, a better approach is to populate the proxy cache using an automated process. This method has been described by J. J. Eksteen and J. P. L. Cloete of

the CSIR in Pretoria, South Africa, in a paper entitled **Enhancing International World Wide Web Access in Mozambique Through the Use of Mirroring and Caching Proxies**. In this paper (available at *http://www.isoc.org/inet97/ans97/cloet.htm*) they describe how the process works:

> "*An automatic process retrieves the site's home page and a specified number of extra pages (by recursively following HTML links on the retrieved pages) through the use of a proxy. Instead of writing the retrieved pages onto the local disk, the mirror process discards the retrieved pages. This is done in order to conserve system resources as well as to avoid possible copyright conflicts. By using the proxy as intermediary, the retrieved pages are guaranteed to be in the cache of the proxy as if a client accessed that page. When a client accesses the retrieved page, it is served from the cache and not over the congested international link. This process can be run in off-peak times in order to maximize bandwidth utilization and not to compete with other access activities.*"

The following command (scheduled to run at night once every day or week) is all that is needed (repeated for every site that needs pre-populating).

```
wget --proxy-on --cache=off --delete after -m http://www.python.org
```

Explanation:

- **-m**: Mirrors the entire site. wget starts at *www.python.org* and follows all hyperlinks, so it downloads all subpages.
- **--proxy-on**: Ensures that wget makes use of the proxy server. This might not be needed in set-ups where a transparent proxy is employed.
- **--cache=off**: Ensures that fresh content is retrieved from the Internet, and not from the local proxy server.
- **--delete after**: Deletes the mirrored copy. The mirrored content remains in the proxy cache if there is sufficient disk space, and the proxy server caching parameters are set up correctly.

In addition, wget has many other options; for example, to supply a password for websites that require them. When using this tool, Squid should be configured with sufficient disk space to contain all the pre-populated sites and more (for normal Squid usage involving pages other than the pre-populated ones). Fortunately, disk space is becoming ever cheaper and disk sizes are far larger than ever before. However, this technique can only be used with a few selected sites. These sites should not be too big for the process to finish before the working day starts, and an eye should be kept on disk space.

Cache hierarchies

When an organization has more than one proxy server, the proxies can share cached information among them. For example, if a web page exists in server A's cache, but not in the cache of server B, a user connected via server B might get the cached object from server A via server B. ***Inter-Cache Protocol*** (***ICP***) and ***Cache Array Routing Protocol*** (***CARP***) can share cache information. CARP is considered the better protocol. Squid supports both protocols, and MS ISA Server supports CARP. For more information, see *http://squid-docs.sourceforge.net/latest/html/c2075.html*. This sharing of cached information reduces bandwidth usage in organizations where more than one proxy is used.

Proxy specifications

On a university campus network, there should be more than one proxy server, both for performance and also for redundancy reasons. With today's cheaper and larger disks, powerful proxy servers can be built, with 50 GB or more disk space allocated to the cache. Disk performance is important, therefore the fastest SCSI disks would perform best (although an IDE based cache is better than none at all). RAID or mirroring is not recommended.

It is also recommended that a separate disk be dedicated to the cache. For example, one disk could be for the cache, and a second for the operating system and cache logging. Squid is designed to use as much RAM as it can get, because when data is retrieved from RAM it is much faster than when it comes from the hard disk. For a campus network, RAM memory should be 1GB or more:

- Apart from the memory required for the operating system and other applications, Squid requires 10 MB of RAM for every 1 GB of disk cache. Therefore, if there is 50 GB of disk space allocated to caching, Squid will require 500 MB extra memory.
- The machine would also require 128 MB for Linux and 128 MB for X-windows.
- Another 256 MB should be added for other applications and in order that everything can run easily. Nothing increases a machine's performance as much as installing a large amount of memory, because this reduces the need to use the hard disk. Memory is thousands of times faster than a hard disk. Modern operating systems keep frequently accessed data in memory if there is enough RAM available. But they use the page file as an extra memory area when they don't have enough RAM.

DNS caching and optimization

Caching-only DNS servers are not authoritative for any domains, but rather just cache results from queries asked of them by clients. Just like a proxy server that caches popular web pages for a certain time, DNS addresses are cached until their ***time to live*** (***TTL***) expires. This will reduce the amount of DNS traffic on your Internet connection, as the DNS cache may be able to satisfy many of the queries locally. Of course, client computers must be configured to use the caching-only name server as their DNS server. When all clients use this server as their primary DNS server, it will quickly populate a cache of IP addresses to names, so that previously requested names can quickly be resolved. DNS servers that are authoritative for a domain also act as cache name-address mappings of hosts resolved by them.

Bind (named)

Bind is the de facto standard program used for name service on the Internet. When Bind is installed and running, it will act as a caching server (no further configuration is necessary). Bind can be installed from a package such as a Debian package or an RPM. Installing from a package is usually the easiest method. In Debian, type

```
apt-get install bind9
```

In addition to running a cache, Bind can also host authoritative zones, act as a slave to authoritative zones, implement split horizon, and just about everything else that is possible with DNS.

dnsmasq

One alternative caching DNS server is ***dnsmasq***. It is available for BSD and most Linux distributions, or from *http://freshmeat.net/projects/dnsmasq/*. The big advantage of dnsmasq is flexibility: it easily acts as both a caching DNS proxy and an authoritative source for hosts and domains, without complicated zone file configuration. Updates can be made to zone data without even restarting the service. It can also serve as a DHCP server, and will integrate DNS service with DHCP host requests. It is very lightweight, stable, and extremely flexible. Bind is likely a better choice for very large networks (more than a couple of hundred nodes), but the simplicity and flexibility of dnsmasq makes it attractive for small to medium sized networks.

Windows NT

To install the DNS service on Windows NT4: select Control Panel ➜ Network ➜ Services ➜ Add ➜ Microsoft DNS server. Insert the Windows NT4 CD

when prompted. Configuring a caching-only server in NT is described in Knowledge Base article 167234. From the article:

> "*Simply install DNS and run the Domain Name System Manager. Click on DNS in the menu, select New Server, and type in the IP address of your computer where you have installed DNS. You now have a caching-only DNS server.*"

Windows 2000

Install DNS service: Start ➜ Settings ➜ Control Panel ➜ Add/Remove Software. In Add/Remove Windows Components, select Components ➜ Networking Services ➜ Details ➜ Domain Name System (DNS). Then start the DNS MMC (Start ➜ Programs ➜ Administrative Tools ➜ DNS) From the Action menu select "Connect To Computer..." In the Select Target Computer window, enable "The following computer:" and enter the name of a DNS server you want to cache. If there is a . [dot] in the DNS manager (this appears by default), this means that the DNS server thinks it is the root DNS server of the Internet. It is certainly not. Delete the . [dot] for anything to work.

Split DNS and a mirrored server

The aim of split DNS (also known as ***split horizon***) is to present a different view of your domain to the inside and outside worlds. There is more than one way to do split DNS; but for security reasons, it's recommended that you have two separate internal and external content DNS servers (each with different databases).

Split DNS can enable clients from a campus network to resolve IP addresses for the campus domain to local RFC1918 IP addresses, while the rest of the Internet resolves the same names to different IP addresses. This is achieved by having two zones on two different DNS servers for the same domain.

One of the zones is used by internal network clients and the other by users on the Internet. For example, in the network below the user on the Makerere campus gets *http://www.makerere.ac.ug/* resolved to 172.16.16.21, whereas a user elsewhere on the Internet gets it resolved to 195.171.16.13.

The DNS server on the campus in the above diagram has a zone file for *makerere.ac.ug* and is configured as if it is authoritative for that domain. In addition, it serves as the DNS caching server for the Makerere campus, and all computers on the campus are configured to use it as their DNS server.

The DNS records for the campus DNS server would look like this:

```
makerere.ac.ug
www     CNAME   webserver.makerere.ac.ug
ftp     CNAME   ftpserver.makerere.ac.ug
mail    CNAME   exchange.makerere.ac.ug
mailserver      A       172.16.16.21
webserver       A       172.16.16.21
ftpserver       A       172.16.16.21
```

But there is another DNS server on the Internet that is actually authoritative for the *makerere.ac.ug* domain. The DNS records for this external zone would look like this:

```
makerere.ac.ug
www     A 195.171.16.13
ftp     A 195.171.16.13
mail    A 16.132.33.21
        MX mail.makerere.ac.ug
```

Split DNS is not dependent on using RFC 1918 addresses. An African ISP might, for example, host websites on behalf of a university but also mirror those same websites in Europe. Whenever clients of that ISP access the website, it gets the IP address at the African ISP, and so the traffic stays in the same country. When visitors from other countries access that website, they get the IP address of the mirrored web server in Europe. In this way, international visitors do not congest the ISP's VSAT connection when visiting the university's website. This is becoming an attractive solution, as web hosting close to the Internet backbone has become very cheap.

Internet link optimization

As mentioned earlier, network throughput of up to 22Mbps can be achieved by using standard, unlicensed 802.11g wireless gear. This amount of bandwidth will likely be at least an order of magnitude higher than that provided by your Internet link, and should be able to comfortably support many simultaneous Internet users.

But if your primary Internet connection is through a VSAT link, you will encounter some performance issues if you rely on default TCP/IP parameters. By optimizing your VSAT link, you can significantly improve response times when accessing Internet hosts.

TCP/IP factors over a satellite connection

A VSAT is often referred to as a ***long fat pipe network***. This term refers to factors that affect TCP/IP performance on any network that has relatively large bandwidth, but high latency. Most Internet connections in Africa and other parts of the developing world are via VSAT. Therefore, even if a university gets its connection via an ISP, this section might apply if the ISP's connection is via VSAT. The high latency in satellite networks is due to the long distance to the satellite and the constant speed of light. This distance adds about 520 ms to a packet's round-trip time (RTT), compared to a typical RTT between Europe and the USA of about 140 ms.

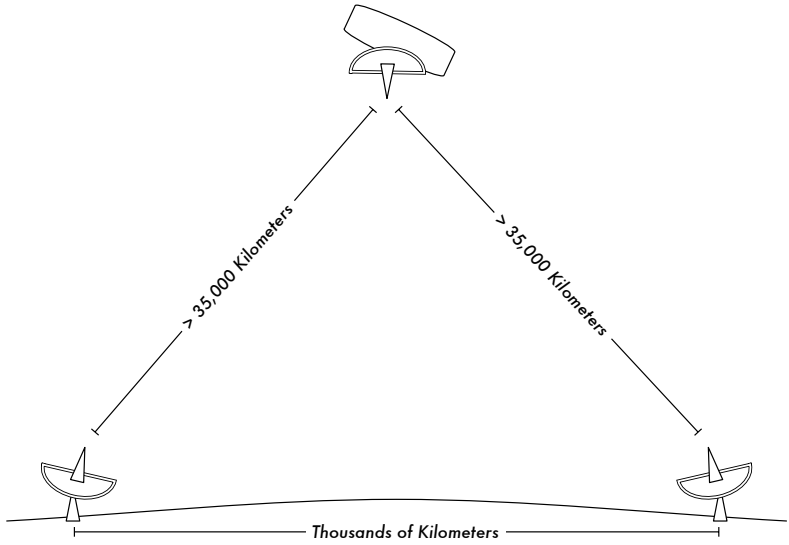

Figure 3.17: Due to the speed of light and long distances involved, a single ping packet can take more than 520ms to be acknowledged over a VSAT link.

The factors that most significantly impact TCP/IP performance are **long RTT**, **large bandwidth delay product**, and **transmission errors**.

Generally speaking, operating systems that support modern TCP/IP implementations should be used in a satellite network. These implementations support the RFC 1323 extensions:

- The ***window scale*** option for supporting large TCP window sizes (larger than 64KB).
- ***Selective acknowledgement*** (***SACK***) to enable faster recovery from transmission errors.
- Timestamps for calculating appropriate RTT and retransmission timeout values for the link in use.

Long round-trip time (RTT)

Satellite links have an average RTT of around 520ms to the first hop. TCP uses the slow-start mechanism at the start of a connection to find the appropriate TCP/IP parameters for that connection. Time spent in the slow-start stage is proportional to the RTT, and for a satellite link it means that TCP stays in slow-start mode for a longer time than would otherwise be the case. This drastically decreases the throughput of short-duration TCP connections. This is can be seen in the way that a small website might take surprisingly long to load, but when a large file is transferred acceptable data rates are achieved after a while.

Furthermore, when packets are lost, TCP enters the congestion-control phase, and owing to the higher RTT, remains in this phase for a longer time, thus reducing the throughput of both short- and long-duration TCP connections.

Large bandwidth-delay product

The amount of data in transit on a link at any point of time is the product of bandwidth and the RTT. Because of the high latency of the satellite link, the bandwidth-delay product is large. TCP/IP allows the remote host to send a certain amount of data in advance without acknowledgment. An acknowledgment is usually required for all incoming data on a TCP/IP connection. However, the remote host is always allowed to send a certain amount of data without acknowledgment, which is important to achieve a good transfer rate on large bandwidth-delay product connections. This amount of data is called the ***TCP window size***. The window size is usually 64KB in modern TCP/IP implementations.

On satellite networks, the value of the bandwidth-delay product is important. To utilize the link fully, the window size of the connection should be equal to the bandwidth-delay product. If the largest window size allowed is 64KB, the maximum theoretical throughput achievable via satellite is (window size) / RTT, or 64KB / 520 ms. This gives a maximum data rate of 123KB/s, which is 984 Kbps, regardless of the fact that the capacity of the link may be much greater.

Each TCP segment header contains a field called ***advertised window***, which specifies how many additional bytes of data the receiver is prepared to accept. The advertised window is the receiver's current available buffer size. The sender is not allowed to send more bytes than the advertised window. To maximize performance, the sender should set its send buffer size and the receiver should set its receive buffer size to no less than the bandwidth-delay

product. This buffer size has a maximum value of 64KB in most modern TCP/IP implementations.

To overcome the problem of TCP/IP stacks from operating systems that don't increase the window size beyond 64KB, a technique known as ***TCP acknowledgment spoofing*** can be used (see Performance Enhancing Proxy, below).

Transmission errors

In older TCP/IP implementations, packet loss is always considered to have been caused by congestion (as opposed to link errors). When this happens, TCP performs congestion avoidance, requiring three duplicate ACKs or slow start in the case of a timeout. Because of the long RTT value, once this congestion-control phase is started, TCP/IP on satellite links will take a longer time to return to the previous throughput level. Therefore errors on a satellite link have a more serious effect on the performance of TCP than over low latency links. To overcome this limitation, mechanisms such as ***Selective Acknowledgment*** (***SACK***) have been developed. SACK specifies exactly those packets that have been received, allowing the sender to retransmit only those segments that are missing because of link errors.

The Microsoft Windows 2000 TCP/IP Implementation Details White Paper states

> "*Windows 2000 introduces support for an important performance feature known as Selective Acknowledgment (SACK). SACK is especially important for connections using large TCP window sizes.*"

SACK has been a standard feature in Linux and BSD kernels for quite some time. Be sure that your Internet router and your ISP's remote side both support SACK.

Implications for universities

If a site has a 512 Kbps connection to the Internet, the default TCP/IP settings are likely sufficient, because a 64 KB window size can fill up to 984 Kbps. But if the university has more than 984 Kbps, it might in some cases not get the full bandwidth of the available link due to the "long fat pipe network" factors discussed above. What these factors really imply is that they prevent a single machine from filling the entire bandwidth. This is not a bad thing during the day, because many people are using the bandwidth. But if, for example, there are large scheduled downloads at night, the administrator might want those downloads to make use of the full bandwidth, and the "long fat pipe network" factors might be an obstacle. This may also become critical

if a significant amount of your network traffic routes through a single tunnel or VPN connection to the other end of the VSAT link.

Administrators might consider taking steps to ensure that the full bandwidth can be achieved by tuning their TCP/IP settings. If a university has implemented a network where all traffic has to go through the proxy (enforced by network layout), then the only machines that make connections to the Internet will be the proxy and mail servers.

For more information, see *http://www.psc.edu/networking/perf_tune.html* .

Performance-enhancing proxy (PEP)

The idea of a Performance-enhancing proxy is described in RFC 3135 (see *http://www.ietf.org/rfc/rfc3135*), and would be a proxy server with a large disk cache that has RFC 1323 extensions, among other features. A laptop has a TCP session with the PEP at the ISP. That PEP, and the one at the satellite provider, communicate using a different TCP session or even their own proprietary protocol. The PEP at the satellite provider gets the files from the web server. In this way, the TCP session is split, and thus the link characteristics that affect protocol performance (long fat pipe factors) are overcome (by TCP acknowledgment spoofing, for example). Additionally, the PEP makes use of proxying and pre-fetching to accelerate web access further.

Such a system can be built from scratch using Squid, for example, or purchased "off the shelf" from a number of vendors.