Toward a Practical and Seamless Wireless Backbone

Independent Study, 600.702.12, Dr. Yair Amir

Michael Hilsdale

May 17, 2004

Initial Project Description

Investigation will focus on connecting computers to wireless backbone networks with minimal, if any, custom software installed on the clients. Goals include permitting a computer to transparently connect to a backbone node, routing its traffic through that node onto the backbone network, and maintain connectivity while the client's view of the backbone network changes. Work will begin by first enumerating the technical requirements to fulfill the goals. Then, specific issues from that list will be identified for implementation during the semester.

Description on file with registrar
as provided at beginning of semester

Introduction

The driving focus of this project has been to permit off-the-shelf mobile client nodes to operate with Internet connectivity in a wireless backbone network *without the need for custom software*. Configuration is permissible (such as specifying which wireless network to associate with), but *nothing other than that which is commonly available to clients should be required for connection*. An added benefit of this goal is the lack of any platform dependent solution; any DHCP enabled wireless client is able to benefit from the backbone network.

This semester was an analysis of wireless backbone networks in which clients connect to access points which communicate with peers to pass traffic to Internet-connected gateways. Environmental issues to cope with include mobile client nodes, mobile backbone nodes, and temporary local severances of connectivity. As a starting platform for this work, I made use of the Spines[1] overlay network software.

Project Goal

The close proximity of access point nodes in a deployed area should be leveraged to permit intra-node communication, thereby eliminating the requirement for each access point to be directly wired for network connectivity. Client nodes should be able to connect to whichever access point they are in range of, and that access point should route all clients' traffic to other access point nodes until that traffic is routed onto the Internet. Additionally, any incoming traffic from the Internet should be routed through the network to the appropriate access point which has direct contact with the client recipient of the traffic. Changes to the topology of the access point network should be automatically detected and compensated for, and client nodes should be permitted to roam the network's coverage area while maintaining full Internet connectivity.

[1] Spines, by Claudiu Danilov and Yair Amir, http://www.spines.org/

Background

Wireless Networks

Current commodity IEEE 802.11 wireless radio networks operate in one of two modes. The majority work in "infrastructure" mode in which client computers (laptops, PDAs, wireless desktops, etc.) connect to a central access point for Internet connectivity. The access point is directly connected to a hard-wired Ethernet connection, and all inter-client communication must go through the access point. The implementation and deployment simplicity of this hub-and-spoke topology is perhaps its greatest strength, as evidenced by the widespread use of personal and corporate wireless networks.

The second mode of operation is "ad-hoc" in which all participating nodes are peers, and inter-node connectivity is established by direct connections. Routing is not automatically provided to ad-hoc networks; if a node is not within the direct range of another, the two cannot communicate with each other even if they share a common neighbor or chain of neighbors. Internet connectivity, if available at all, is typically provided by a single client node which happens to have a direct connection. Any nodes requiring Internet access must be within communication range of this node. For the past several years, ad-hoc networks have been the focus of intense research.

Internet connectivity in infrastructure mode networks is provided by the access point which typically assigns IP addresses to clients using DHCP, specifying itself as the gateway. Alternatively, the access point can serve as a bridge, thereby permitting a wired host to handle authentication, IP assignment, and routing services. Internet connectivity in ad-hoc networks is not standardized, though is typically implemented by a single designated node acting as a DHCP server and gateway.

The optimal layout of access points is itself a field of considerable study. Due to FCC regulations, access points are limited in the amount of power with which they can transmit. Additionally, client nodes commonly have limited battery power sources which must be conserved whenever possible. Plus, a multiple cell wireless network is much preferred over a more centralized network to avoid widespread interference to all nodes in range. These requirements necessitate the use of multiple access points to provide sufficient coverage for a target area. The interesting side effect is that most access points are within communication range of other access points, an advantage that is not commonly leveraged. A negative consequence to their close proximity is the likelihood for interference between them.

In addition to requiring an electrical connection, infrastructure access points inherently need to be directly attached to a wired network. Installing a new wireless network in a building can be costly. Ethernet cable needs to be routed through walls to each deployed access point, which are typically located in high locations to permit better radio wave propagation. This problem is compounded when attempting to run cables through older structures. While even installing simple electrical cabling is often not a trivial task, there is a definite advantage to only having to run one power cable and to not run an additional cable for the network. In outdoor scenarios, it would be possible to eliminate even the electrical wiring requirement by installing solar powered wireless nodes with a rechargeable battery backup.

Spines

Spines is an overlay network package which maintains routing information between member nodes. With simple modifications[2], Spines can be used to automatically detect the presence of other Spines nodes and add them to the routable Spines network. When nodes loose connectivity to their neighbors, routing information in the network is updated and propagated throughout the network. Each Spines node maintains a link-state view of the entire network.

DHCP

Assignment

The Dynamic Host Control Protocol (DHCP) is commonly used to automatically assign IP address to client computers. To request an IP lease, a four-way handshake takes place (see Figure 1).

A client sends out a UDP Discover message looking for any available DHCP server. Each server receiving the Discover message sends an Offer message indicating that a specified IP address is available for use. The client arbitrarily chooses amongst the available Offer messages and sends a Request response destined for the specific server with which it would like a lease. The server, having received a Request message for a promised IP address issues an ACK message notifying the client that the IP is indeed available for use. Only once the ACK has been received by the client can the client begin to use the assigned IP address.

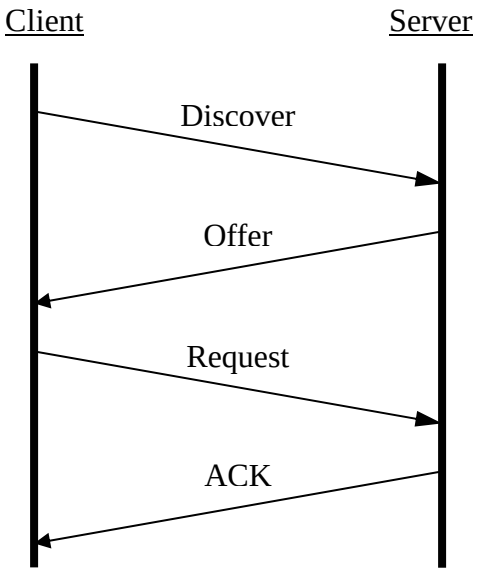

Figure 1
DHCP Four-way handshake

If a client does not have an assigned IP address, all IP packets are sent with a source address of 0.0.0.0 to a broadcast destination of 255.255.255.255.

With the Offer and ACK messages, the server notifies the client about the length of the lease (specified in seconds by a 32-bit integer), along with the gateway for the client to use, DNS servers, and subnet mask. Included as part of each DHCP exchange is a client-assigned transaction ID which permits the tracking of specific requests. The client also provides its globally unique, manufactured-defined Machine Address Code (MAC) address as part of the request. Servers are required to use the client's MAC address as a unique identifier.

Renewal

Half-way during a client's lease, the client will attempt to renew its lease by sending a unicast renewal Request to the previously contacted DHCP server. The client will continue sending out Request messages to the server until it receives an ACK. If no ACK is received, the client will

[2] By itself, Spines does not automatically detect neighbors; they must be specified upon initialization.

begin sending out broadcast Discover messages looking for any available DHCP server to respond.

Under no circumstances should a client continue to use an assigned IP address once its lease has expired. If the client is unable to acquire a new lease by that time, it must forfeit its IP and continue to send Discover messages until either a response is received or a timeout is reached.

System Design

A wireless backbone network consists of a collection of access point nodes which can communicate with each other wirelessly. Some nodes are connected to a wired Internet connection, and client computers connect to whichever access point is closest to them for Internet access. The node that a client connects to is responsible for ensuring that all Internet-bound traffic sent by the client is routed to an access point which has a direct Internet connection.

If neighboring access points loose connectivity with each other, the rest of the access points in the network need to know about the update. Likewise, if two access points are able to establish a connection, the network must be informed. Changes in connectivity cause changes in the routing of the network. Each node knows the full topology of the network. If a node needs to route a packet to a specific node or if it needs to find which the nearest Internet-connected node is, it uses its last known view of the network. Thus, any change in connectivity in the network needs to be promptly discovered and communicated to all nodes in a timely manner in order for them to have up-to-date knowledge of the network. For this, the Spines package is used.

Spines maintains current global routing information throughout network topology changes. An auto-discovery component has been added which permits neighboring Spines nodes to realize the potential for a connection and to begin the link initialization procedure. This is implemented through the use of regular broadcast packets that each node sends. When a broadcast packet is received by a node, that node will attempt to establish a link with the node that sent the beacon. This self-organizing and self-maintaining network is vital to the wireless backbone goal.

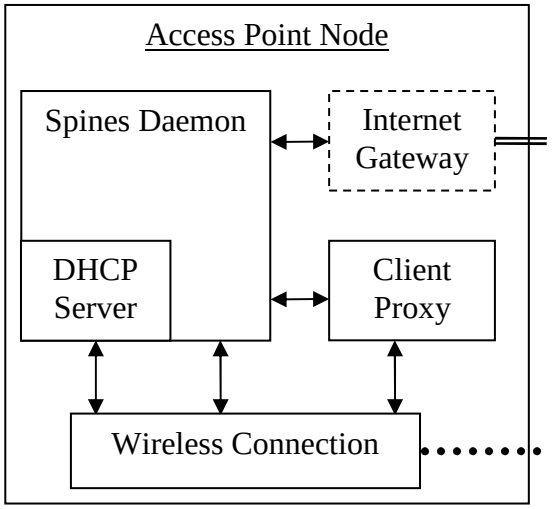

Figure 2
Access Point Components

Each Spines node also has a built-in DHCP server. Clients that connect to it will negotiate a lease with the access point, and the node will assign the client an IP address that is unique to the network. The DHCP server will also instruct the client to use itself as a gateway. After a client confirms its acceptance of an offered DHCP lease, the node will notify the rest of the Spines network of the client's connection. A client need only support the DHCP protocol in order to identify which access point it is in range of and to begin using that node for network connectivity.

All nodes can act as routers for client Internet traffic.[3] Once a DHCP server directs a client to use a node as a gateway, that node must be able to receive traffic and ensure that it is forwarded to the Internet. A proxy host is installed on each node which receives traffic sent by the client destined for the Internet. The proxy then redirects the traffic to the local Spines daemon which forwards the packets to the nearest (fewest hops) Internet-connected node. When a Spines daemon on an Internet-connected node receives traffic destined for the outside world, Spines redirects that traffic to a locally running gateway server which manages the NAT mappings and actually sends the packets out onto the Internet. The same gateway server also manages incoming traffic, mapping inbound packets to the proper private client IP address and then encapsulating the traffic by sending it back to the Spines daemon for delivery through the Spines network. A Spines node that receives traffic destined for a client will redirect that traffic to the local proxy server which then sends the packets directly to the client.

Use of DHCP for Client Discovery and Registration

Spines is an excellent platform for maintaining routing information between hosts on the network. Links between nodes are kept alive by periodic hello messages that are acknowledged by the host on the opposite end of the link. After initialization, a Spines node need only respond to the regular pings in order for it to remain registered on the network. The DHCP solution was encountered by realizing this fact. A node need only send regular pings to a Spines node in order to be registered in Spines. By connecting the link initialization and the hello protocol to the custom DHCP server, client nodes that are not even running the Spines software may unknowingly realize the benefits of being globally registered and routable to on the Spines network. By issuing a short lease time, Spines nodes can promptly know when a client is no longer within range by noting the lack of hello traffic from that node, and then they could take appropriate action by notifying the rest of the Spines network of the disconnection.

[3] The work described in this paragraph was contributed by JHU graduate student Nilo Rivera. The gateway server described uses the LIBPCAP software (http://www.tcpdump.org/).

Network Topology

Below is a sample Spines wireless backbone topology.

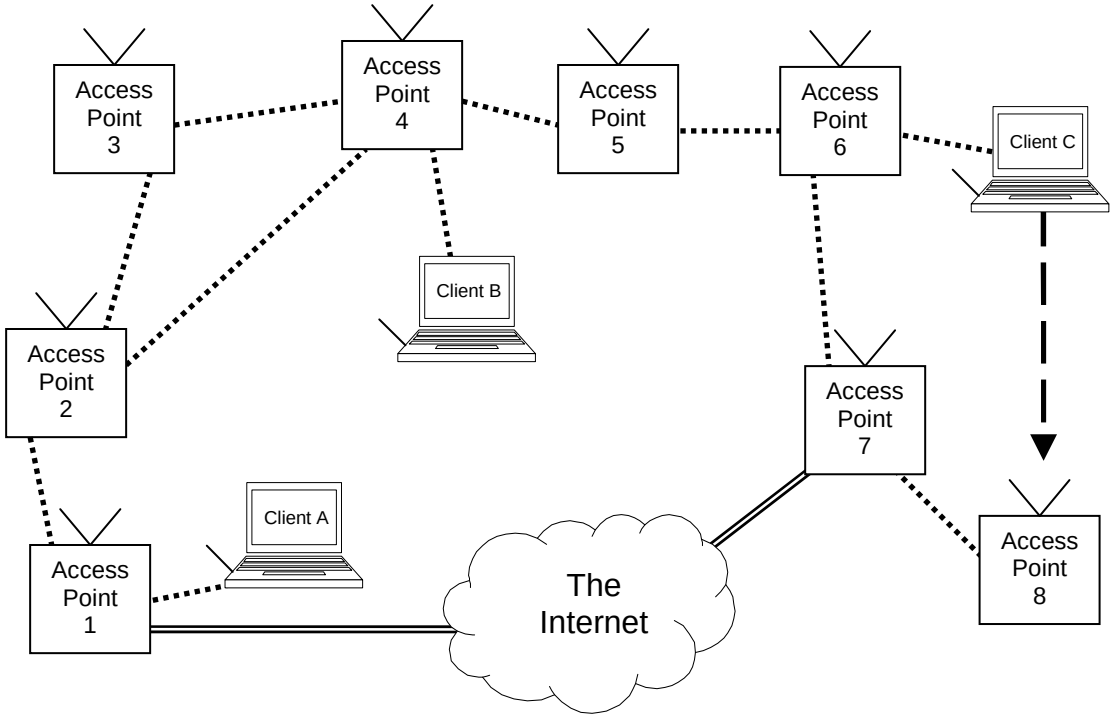

Figure 3
Typical wireless backbone topology

Access points 1 and 7 are Internet-connected nodes and are thus able to serve as gateways for the wireless network. Clients A and B use access point 1 for Internet connectivity, while client C uses access point 7 throughout its journey.

This diagram highlights several capabilities of the network. Of particular interest:

- Note that as Client C moves through the coverage area, it will eventually move from within range of access point 6, to 7, and then to 8, likely reaching points in which it is in the range of more than one node. The client will arbitrarily choose a node to associate with, and the network will automatically notify the other nodes which access point the client is connected to.
- If the link between access points 3 and 4 were severed, the redundancy of the network will permit traffic to continue to flow.
- If the link between access points 1 and 2 were severed, then Internet connectivity for client B would be automatically rerouted to go through access point 7.

Implementation Considerations

Below follows a description of the various technical challenges of the system and their associated solutions.

1. *Access points need to be able to communicate with each other in addition to client nodes.*

   This is accomplished by operating all access points and client nodes in ad-hoc mode, permitting a purely peer-based network layer along with the use of custom routing. All wireless nodes in the network are configured for the same wireless channel, the same SSID network name (I used "mist"), and the same WEP encryption setting (if used at all).

2. *Spines is used to facilitate connectivity between the access points. Access point nodes which loose connectivity to neighbors must be able to report this to Spines so that the global routing information can be updated. Additionally, access points which realize new connectivity to other access points should automatically establish a link and notify the rest of the Spines network of the new routing.*

   The Spines package handles the automatic propagation of routing information to each connected Spines node. When an established link is severed (resulting from temporary radio interference, a localized node failure, or a mobile node), Spines already notifies the remaining network of the disconnection, and each node calculates new global routing knowledge.

   Spines does not, however, provide a means for automatically establishing links with previously unknown nodes. This functionality has been added to Spines through the use of regular broadcast beacons which announce the presence of a Spines node. If a beacon is heard by a Spines node, the receiving node attempts to establish a connection with the broadcasting node by sending a hello message. If the beaconing node is able to receive packets from that node[4], then a connection is established, the new link is announced to the Spines network, and updated global routing information is computed.

3. *Clients need to be assigned unique IP addresses that do not conflict with other clients in the network. This also permits Spines to uniquely identify the nodes for routing purposes.*

   Through the use of DHCP, clients request are assigned IP addresses on the network which are unique[5]. This permits client nodes to roam throughout the coverage area without encountering another ad-hoc node with a conflicting IP address.

4. *Clients associate themselves with access point nodes. The Spines network needs to be apprised of this registration so that Spines knows how to route incoming traffic back to the clients from the Internet.*

[4] Two-way connectivity is not guaranteed in wireless networks. Due to radio wave propagation effects or unequal radio equipment (different antennas, transmission power, and/or receiver sensitivity), it is possible to have one-way links. While it would be theoretically possible to leverage these one-way links in a routing protocol, this issue remains unaddressed in Spines and in this project.

[5] The current version of the software does not assign globally unique IP addresses to clients, though that functionality could be added with a minor code change. Instead of each node issuing DHCP leases for private IPs in the range of 10.0.0.100+, they could assign IPs of the form 10.0.R.100+, where R is the last byte of the IP address of the issuing router. Additionally, the DHCP server would provide a subnet mask of 255.255.0.0 to clients.

The custom DHCP server built into each access point uses the client's globally unique MAC address as an identifier so that other Spines nodes can uniquely identify the nodes throughout the DHCP registration process. Once a client has been issued a DHCP lease, the access point notifies the other Spines nodes of the existence of a new node. The client's MAC address is included as part of the usual Spines node data structures. As far as Spines is concerned, the new node is just like any other node, with the exception of the presence of a MAC address and a flag which indicates that this is a client node. This permits the use of the already-provided routing infrastructure of Spines to route Internet traffic back to this node. (See Significant Discoveries below.) DHCP leases also direct the clients to use the issuing access point as their gateway.

5. *Clients must be able to roam within the coverage area without manual reconfiguration. Client nodes may be within range of multiple access points, and they may randomly loose connectivity to any given access point. A client should always have an Internet connection as long as the client is within the range of at least one Spines node which has transitive connectivity to the Internet.*

   Through the use of a short DHCP lease time of one minute[6], clients are continually requesting a renewal of their IP lease. The DHCP protocol requires that clients attempt to renew their lease halfway during the lease time to ensure that the IP will be available for continued use once the lease has expired. This is in essence used as a hello protocol for the client nodes. The clients regularly send DHCP packets (effectively pings) to the access point with which they are associated, and that access point permits the renewal of the IP lease. This notifies the access point that the client is indeed still within range and is relying upon the access point for network connectivity.

   If an access point fails to hear a renewal request from a client node within the agreed upon time frame, the access point can notify the rest of the Spines network of the absence of a direct link to that client.

   The DHCP protocol also specifies that if a client is unable to renew an existing lease, the client should send out a discover packet that should be answered by any receiving DHCP server. Whenever an access point receives a DHCP discover packet from a client, it will respond back with an offer. If the client is within the range of multiple access points, it should receive multiple offers. The client chooses an offer and notifies the issuing server of its acceptance.

   If a client fails its renewal process and is not able to hear back from its original lease-issuing server, its subsequent discover packets will be heard by any other access point within range. The client will associate with the new access point, be granted a lease extending the use of its existing IP address, be directed to use the new access point as its gateway, and the DHCP server will notify the other Spines access points that the client has moved from its previous location to the new one, thereby causing a global routing information update.

6. *Access points receiving traffic destined for the Internet must know how to route that traffic to the appropriate Internet-connected access point.*

[6] The one minute lease time has been used for development. It would be possible to use a shorter lease time so that topology changes would be more quickly recognized.

When any access point receives traffic from a client that is destined for the Internet, the access point must know how to route the traffic. Trivially, if that access point is itself connected to the Internet, it will forward the traffic to the Internet-connected interface. Otherwise, the access point needs to reference the global Spines routing knowledge to identify the closest node that has an Internet connection and then route the traffic to that node. Upon initialization of the Spines daemon, nodes are designated as Internet-connected or not. This information is added to the Spines routing information which all other nodes know about, thereby spreading knowledge about which nodes have Internet connectivity.

7. *Return traffic entering the Spines network from the Internet must be routed back to the appropriate client node.*

   An Internet connected access point will have an IP address that is in a separate subnet from those of the Spines nodes and clients. All outbound network traffic that passes through that access point will have a source address of that access point's Internet-connected interface. At this point, the access point is acting like an ordinary Network Address Translation (NAT) gateway, requiring it to maintain a table mapping outbound traffic requests to inbound client IPs. Any responses to the outbound traffic will be targeted to the IP address of the access point. The access point must then use its NAT lookup table to determine to which internal client IP the response needs to be routed.

   This is one of the strengths of the system's design. If a client has moved from one access point node to another, the Spines network will have been notified of the new reassociation and can route the message along the new path. There is a timeout issue, but it is addressed in the following point.

8. *Ideally, clients who roam between access points should be able to maintain any active socket connections. Additionally, the lag time between reassociating with a new access point should be minimized.*

   When a client roams to a new access point, its new presence will not be noticed until the client's DHCP renew requests time out and the client requests a new IP through a discover message. Once a new access point establishes a successful four-way handshake with the client, the rest of the Spines network can be notified of the change.

   There is a potential here for a significant delay before the updated topology can be applied. The client may continue to try to route traffic through the old access point even though it is no longer within range of it. The client could potentially wait as long as *L* time (where *L* is the lease time) before realizing a new connection[7]. If a sufficiently short lease time is chosen, then even TCP connections (with a timeout of one minute) should be able to be maintained. Any active UDP connections, which inherently provide no delivery guarantees, would revert to any application-level retransmission protocol.

   If a client node has Internet connectivity via a routed path in the Spines network through a designated Internet-connected Spines node, all outbound traffic to the Internet will have

[7] Assume that a client will successfully renew its upcoming lease halfway through its previous lease. Then, halfway through its new lease (*L* time later), it attempts to contact the issuing DHCP server for a lease renewal. Unable to do so, it will attempt to acquire a new lease from any available DHCP server. Assuming that the client never left the range of *some* access point, it should receive a prompt reply to its discover message.

a source IP of that Internet-connected node. The client can continue to roam throughout the Spines network, and as it roams it will reassociate with a new access point. However, if the new access point that it associates with no longer has an optimal path to the previous node that was used for Internet connectivity, any new outbound Internet traffic will have a source address of whatever new Internet-connected node is available to the new access point. As is, this is a potential problem since socket connections would be broken. Even if no sockets are currently open, applications that have active session with external hosts may not behave as expected if their source IP changes[8].

If a client is able to renew its lease within a time that is acceptable to TCP timeouts, the socket connection should remain open. This, however, has not yet been fully tested. It is possible that the client operating system, upon reconfiguring its gateway address, may reset all local socket connections, causing an immediate disconnection once the new lease is applied (even if the same client IP address is used).

9. *It also would be desirable for the system to support inter-client communications. If one client wishes to send packets to another client, it should be possible to contact the target client even if the two are not within direct range. As long as two clients are transitively connected, they should be able to communicate with each other.*

   If a client needs to connect to another client, it should be capable of doing so. In ad-hoc mode, this is provided automatically. If two hosts are within range of each other, they can communicate directly. However, if two client nodes are connected to different Spines-connected access points, they would have to route traffic through the Spines network to enable intercommunication.

   Though unimplemented in this project, there are some potential solutions to this problem. One would involve simply setting the subnet mask for all client nodes to 255.255.255.255 and delegating all routing activity to the access points. This would be a relatively easy solution to implement. The main (though possibly minor) drawback would be the inability for client nodes to communicate directly with each other. Instead, inter-client communication would be identical to that which is currently employed under infrastructure mode networks.

   A second (albeit more complex) option would require some smart handling by a local access point. If two clients are both connected to the same access point, then that access point is aware of this fact. The clients, having a subnet mask that permits them to do so, would simply communicate with each other without the need to loop through the access point. If two clients were connected to different Spines-connected access points, however, then the Spines network (but not the clients) would know that traffic between the two would need to be routed. Whenever an IP-enabled host attempts to send information to another, its Address Resolution Protocol (ARP) table is consulted to see if the MAC address of the destination host is in its table. (ARP tables are consulted for destination IPs only if the target host is within the configured subnet. Otherwise, traffic

[8] An example of this would be a client using a secure website requiring authentication which ties into the source IP. Even if the client closes its socket connection after requesting a page, subsequent requests to the server will have a different source IP than that which was used upon authentication, which may cause an authentication failure. This effect is often seen in networks which use externally rotating (load balancing) proxy servers to handle outbound Internet traffic.

is sent to the specified gateway for routing. Peer clients would naturally be in the same subnet.) If the table does not contain an entry, the client sends an ARP message to request the MAC address of the desired host. Normally, unless the destination host is within the same subnet, the ARP request would go unanswered and would time out, preventing any communication. One solution to this problem would call for the access point to recognize this scenario and to send out specially-crafted ARP packets to spoof the sending client into sending traffic to the access point directly instead. The access point, listening promiscuously, would then pick up the packet and deliver it to the destination client by routing it through the Spines network.

Future Work

Linksys routers as a development platform

The Linksys routers were initially purchased with the plan to use them simply for 802.11g wireless connectivity. While researching them, I noticed that GPL source code for the routers was freely available from the manufacturer's website (see Appendix A). Further investigation lead to the realization that many open source groups were making modifications to the router's firmware and providing those modifications on the Internet. The Linksys routers, costing <$80 are far cheaper than the full-powered $500+ mini-ITX computers used for development.

This enables the purchase of a considerably large number of cheap routers for deployment, each of which can function as a full Spines access point node as described in this document. Modifications to the source code along with the custom Spines and DHCP code could be posted to a website, along with a stable firmware image. Now, anyone with the most popular wireless router could simply download a new firmware image and have immediate access to the benefits of self-organizing wireless backbone networks!

Infrastructure Operation

The current project uses ad-hoc communication between wireless nodes. The most common configuration in use today is infrastructure mode, although virtually all cards support ad-hoc operation. It would be possible for this work to make use of the infrastructure mode of operation in access points, with the use of the Wireless Distribution System (WDS) protocol. WDS permits access points to be able to communicate both with other access points and with client nodes at the same time without requiring a reassociation delay. WDS is not commonly used, but the implementations that do exist require a static definition of the MAC addresses of up to six neighboring access points (though not practical due to interference concerns). The static definition requirement limits the mobility potential of the access points. Future work could modify a wireless driver's WDS implementation to permit the dynamic addition of access point MAC addresses, possibly permit the use of more than six, and (more importantly) permit the auto-discovery of previously unknown access points.

Given that modern wireless configuration tools built into operating systems permit the use of ad-hoc networks as easily as they do infrastructure mode access points, this may in fact not be a significant concern.

Appendix A: Linksys Routers

The Linksys WRT54G routers are 802.11g wireless access points with a built-in four-port switch and a Wide Area Network (WAN) port. Currently the most popular wireless router, their reasonable price and robust performance make them great additions to home or office networks.

Under the hood, these MIPS-based machines run a trimmed down version of Linux. All of the source code used on router, including a cross compiler and documentation, is available from Linksys.

The five routers that we purchased are v2 hardware routers. Their hardware consists of a 200 MHz MIPS-based Broadcom chip, 16 MB of RAM, and 4 MB of flash ROM for the firmware. Configuration settings are stored in separate NVRAM.

Here are four options for working with the WRT54Gs:

- Use the BatBox distribution to get shell access. This is the quickest and safest way to immediately jump head-first into playing with the routers.
- Download and run the Sveasoft firmware. This is the quickest way to get lots of useful features into the router's firmware, including telnet and SSH support.
- Download and recompile the Sveasoft firmware to suite your needs.
- Download and recompile the Linksys source code to modify as you wish. This may be the more difficult option since you're working from a fresh copy of the source code and you'll need to manually add whatever functionality that is not already provided in the stock setup. On the other hand, this is the cleanest way to add code without any unwanted extra stuff.

Whichever development option is chosen, it is important to remember that *you do not need to recompile and reflash the firmware just to run a new binary*. You can always upload your new binaries to the ramdisk in /tmp, chmod +x the binary, and execute it. Technically, flash ROM has a limited reflash life, and it is always a procedure with some level of risk.

Compilation Guidelines

To compile anything for the WRT54Gs (including any of the firmware packages), you first need to obtain a cross compiler. Linksys includes a precompiled cross-compiler with their source code distribution. You should be able to use the binaries as-provided, but you might need to recompile the GCC cross-compiler. Their directions for doing this are rather straightforward, but it sure does take a long time. In either case, the final cross-compiler should be installed (as root) to a common location on the computer that you will compile MIPS code. They suggest that you install it to /opt/brcm/, and you'll need to follow their directions and add the two paths to the compiler executables to your PATH environment variable. One is the hndtools-mipsel-linux compiler, the other is the hndtools-mipsel-uclibc compiler.

In addition to installing and adding the compiler directories to your PATH, you'll also need to copy some tools that Linksys provides into a common directory on your target server. From the Linksys distribution, copy the folder WRT54G/release/tools/ to /opt/brcm/, and add it to your PATH variable. In all, there should be three items that need to be added to your path: the two compiler bin directories and the Linksys tools directories.

You should now be set to compile any of the firmware distributions.

To compile an arbitrary program for the WRT54G, you'll need to modify the Makefile to specify the MIPS GCC, CC = mipsel-uclibc-gcc. You'll also need to do this for any other compiler tools you use, such as ld, ar, etc.

Compiling Spines for MIPS

To compile Spines for MIPS, first obtain a recent snapshot from the development CVS server. Next, in the Spines home directory, replace Makefile with Makefile.mips. Then, just run 'make'. That will take care of configuring the stdutil package to use the MIPS compiler in addition to compiling Spines for MIPS. Note that as part of the normal stdutil compilation process, a test is compiled and run to determine whether or not the stdutil can operate on the native platform. This test has been disabled when compiling for MIPS since you cannot execute the target MIPS code in your compilation environment.

WRT54G Resources

Linksys

Linksys website to download firmware source code and cross-compiler:

http://www.linksys.com/support/gpl.asp

The most current official firmware binary is available at

http://www.linksys.com/download/firmware.asp?fwid=201

BatBox

The BatBox distribution is a good starting point to get access to the firmware on the router. It allows you to upload a telnet daemon and a shell so that you can log in. (Properly, it only permits telnet access on the internal interface.) From there, you can browse around and get familiar with the router. Be sure to check out 'help', /proc, 'wl', 'wl status', and 'ifconfig'. The great thing about BatBox is that it leverages a commonly-used buffer overrun vulnerability on the WRT54Gs with the web-based "ping" tool. It then uploads all the files into the ramdisk, with zero risk to the firmware or the NVRAM. When you're done, you just reboot the router, and it's as good as new.

http://www.batbox.org/wrt54g-linux.html

Sveasoft

One of the more mature distributions available is the Sveasoft package. Start there and download their most recent stable firmware. Their package updates the web configuration with some useful configuration options. If you use the Sveasoft distribution, the first thing you should do is go to the Administration web page and enable Boot Wait. This is a failsafe mechanism designed to give you some extra time during router bootup to upload a new firmware if the existing one is corrupt. Search the Sveasoft site for more info on this if you need to.

You can also download their full source code, which you will need if you want to recompile it. They use the same common Linksys code that is publicly available, but they've already ported, included, and configured lots of useful features, and they have a decent web interface for the additions.

The important benefit to the Sveasoft distribution is that their wireless drivers actually support ad-hoc mode, enabling the access point to work like a normal ad-hoc peer and permitting the work described in this paper.

http://www.sveasoft.com/modules/phpBB2/

To enable ad-hoc access with the Sveasoft distribution, log in through the internal Ethernet connection and execute the following commands at the shell prompt:

# wl ap 0	disables the access point
# wl infra 0	turns off infrastructure mode
# wl join mist mode ibss	joins the 'mist' network in ad-hoc mode
# wl wep 0	disables wep encryption

General

Very useful website containing information about the hardware and software on the routers, including references to other open source projects:

http://www.seattlewireless.net/index.cgi/LinksysWrt54g#head-998c4ad7982499899aee917d0f93f3c36f71df0c

Appendix B: iPAQ Compiler Information

To compile for the StrongARM-based iPAQs running PocketPC, you need to download and install several products in a specific order.

Here is a good Microsoft document describing an overview of their mobile developer tools and installation order (see their appendix). Always check it first for up-to-date directions.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnppcgen/html/devtoolsmobileapps.asp

This website should provide current links to the required downloads:

http://www.microsoft.com/windowsmobile/resources/downloads/developer/default.mspx

First install the ActiveSync software that came with the PocketPC. Alternatively, you can probably download it from Microsoft's website.

You'll definitely need to obtain the Microsoft eMbedded Visual C++ 4.0 toolset and associated service packs.

After you download and install eMbedded Visual C++ and any service packs, you need to download and install the PocketPC 2003 SDK.

To compile Spines for the PocketPC, first obtain a recent snapshot from the development CVS server. Then, just browse to the PocketPC directory and open the project file. This will start eMbedded Visual C++, load the project, and permit you to compile. One word of warning: whether you compile for the PocketPC emulator, or whether you compile directly for the StrongARM platform, you have to select your target in *two drop-down locations* in the main eMbedded Visual C++ screen.

Appendix C: Equipment Description

Through the generous allocation of $10,000 from the Engineering Dean, we were able to purchase a variety of equipment for use for this semester's Advanced Distributed Systems course and for my project work. Highlights of the equipment purchased are as follows:

- 6 small mini-ITX computers with 802.11b wireless network adapters. These x86 compatible computers are full systems that are compact in size and suitable for deployment throughout the building. These machines were also used as desktops and development platforms in support of the class.
- 5 iPAQ handheld computers with built-in 802.11b wireless adapters. The small size of the StrongARM-based iPAQs, along with the readily available compilation tools (see Appendix B), made them ideal client nodes.
- 2 laptop computers with built-in 802.11g wireless adapters. The stereotypical mobile node, these laptops were valuable in the development and testing process.
- 5 wireless 802.11g routers. One of the later purchases that we made, these Linux-based routers proved surprising in their potential as a deployment platform (see Appendix A).