Dynamic Host Configuration for Managing Mobility between Public and Private Networks

Allen Miu
MIT Laboratory for Computer Science
aklmiu@lcs.mit.edu

Paramvir Bahl
Microsoft Research
bahl@microsoft.com

Abstract

The usage and service options of a pubic network generally differ from a private (enterprise or home) network and consequently, the two networks are often configured differently. The existence of both types of networks motivates our need to improve support and management of nomadic users who frequently roam between them. We describe a solution that allows client devices to configure themselves dynamically to adapt to the local network configuration. In addition to supporting mobility, we describe how our solution also provides fail-over mechanisms for providing highly available service, load balancing, and location services. Furthermore, our solution can be used to scale networks that are deployed in a large setting. We discuss in detail the various issues that need to be dealt with for achieving true device-level mobility, pointing out several unsolved problems in this area. The algorithms and software proposed in this paper have been implemented, are deployed, and are currently being used in a real-world public network that is operational at the Crossroads Mall in Bellevue, Washington.

1 Introduction

Today, our economy and businesses rely heavily on people having Internet connectivity. This combined with the observation that we have become a highly mobile society in which many of us invariably find ourselves spending a considerable amount of time in public places and at public events compels us to move in the direction of providing high-speed Internet connectivity everywhere we can.

We have built and deployed a *public* wireless network, called the *CHOICE* network (URL: http://www.mschoice.com), which provides individuals Internet access in public places such as shopping malls, airports, libraries, train-stations etc. Our network is based on the widely available IEEE 802.11b standards-based wireless LAN technology [2], which enables us to provide Internet access to authenticated users at speeds up to *25 times* greater than speeds offered by 2.5G and 3G cellular phone networks [1]. Additionally, our network offers policy-based services such as different levels of privacy and security, different amounts of bandwidth, and different location services all on a per-user basis. For the host organization our network provides protection against malicious users and options for detailed accounting and flexible charging. Our design is conducive to developing interesting location services such as location-based buddy lists, electronic in-building navigation, and timely shopping promotions.

The underlying protocol that enables many of the aforementioned features of the CHOICE network is the **P***rotocol for* **A***uthorization and* **N***egotiation of* **S***ervices*, or PANS. PANS is a novel lightweight protocol that facilitates (a) global authentication of users; users can be authenticated from anywhere in the world, (b) authorization, monitoring and management of network access for authenticated users, (c) enforcement of policies on a per-user basis, and (d) device auto-configuration for supporting users who roam between differently configured networks.

Typical usage scenarios for private and public networks are different, and consequently, these networks are generally configured differently. Large corporations tend to be extremely security cautious, taking an enterprise-centric approach where every user is governed by a single policy. User authentication is intended to prevent unknown persons from accessing internal private networks. Public networks are security cautious only to the extent the individual using the network is. The host organization's focus is on establishing the identity of a *previously unknown* user and then giving her access to the network, its resources, and other location services generally for a fee. Hence, tracking who is using the network, what services are being used and how much bandwidth is being used are important. Another difference is, while corporations generally have a high level of confidence and trust in their users (employees), public network operators have to guard against the network users who they might not know well. They need tools to protect themselves from malicious users who are only interested in bringing the network down.

In thinking through the different usage scenarios and studying several privately deployed networks that we know of, we concluded that corporations generally use some sort of a pre-configured shared key mechanism

with hardware encryption to secure network access. Public networks on the other hand perform packet-level processing for both user-level authentication and privacy, and for offering different kinds of services, and keeping track of network use on a per-user basis. Consequently, **client devices have to change behavior according to the network they are accessing**. When accessing the private network (normal mode), the client need not do anything; hardware encryption with a shared key is sufficient to control users' access. However, when accessing the public network (special mode), the client runs through an authentication process and starts using a specialized network access protocol (e.g. PANS), which gets it different types of interesting services.

With these issues in mind, we developed a mobility support mechanism that allows devices to automatically determine how to establish/re-establish network connectivity as roaming users migrate across the different networks.

We present the architecture and operation of the CHOICE network, focusing on the problem of supporting mobility at the device configuration-level. We briefly describe PANS and the features it provides, leaving out details that are documented in [3]. We show how our system's mobility architecture allows us to support other important features like load balancing, scaling, and location services.

The primary contribution of our work is a detailed design of a system and protocol that offers the following important features:

a. Dynamic configuration of client devices, without user intervention, as nomadic users roam between public and private networks.
b. Dynamic configuration extensions that support a fail-over mechanism as well as a scalable key-distribution system, and
c. Support for location services currently not available in other networks.

The rest of this paper is organized as follows: Section 2 sets the stage by describing a typical usage scenario of public and private networks. Section 3 describes the system components of the CHOICE network. Section 4 then articulates the precise mobility problem and in Section 5 and 6, we discuss our design criteria and our solution. In Section 7, we explain how our solution can be extended to help public networks achieve high availability and scalability. We discuss on-going and future work for achieving true mobility in Section 8 and we survey related work in the field in Section 9. Finally, we conclude in Section 10.

2 A Typical Usage Scenario

A person walks into a public place where she has arranged to conduct a business meeting with another person from a different company. Both people are savvy wireless LAN users and come equipped with their notebook computers and wireless LAN cards. The public place has a CHOICE network that is available to the general public for a small fee. As the user sits down at a coffee table waiting for her companion, she switches-on her notebook computer, launches her web browser, and points it to http://choice. If she has not already done so, the user downloads the network access software (PANS client) from the local web server and installs it on her notebook. A reboot of the machine is not required for this installation. Upon installation, the PANS client module detects the presence of the CHOICE network and displays a welcome message to the user indicating to her that she can get Internet access by logging on and establishing her identity. The user then proceeds to authenticate herself via the local organization's log-on page wherein she types in her identity and password. These are sent to a global authentication database to which the local host organization subscribes. When the user's identity is established and authentication granted, the network checks to see the policy that is to be applied for this particular user (e.g. how much bandwidth to give, what security level to grant and how much to charge, default values exists for first time users). Based on policy, the network generates a unique key and sends it to the PANS client. At this point the user's web browser automatically refreshes to the local portal and Internet access is now possible.

After she is done with her meeting she log-offs and the network provides her with some usage statistics. She returns to her company and opens up her notebook, which she had placed in "hibernate" mode. As the notebook turns on, the PANS client senses that a different network is present and stops all special processing that is necessary for the CHOICE network.

We now describe the components of CHOICE and then explain the mobility problem more precisely in the following section.

3 Overview of the Choice Network

The CHOICE network has several system components that manage address allocation, authentication, authorization, security, accounting, and last-hop QoS. Figure 1 illustrates the different components of the CHOICE network as it has been deployed at the Crossroads Mall, Bellevue, Washington. Our description of the CHOICE network will be brief as we refer the reader to [3] for a detailed description of PANS.

3.1.1 Address Management and Naming

The CHOICE network uses a standard DHCP server to lease IP addresses to potential clients. The IP address scope and the lease period are configured by the host organization at setup time. Where DHCP's limited scope

is an issue, a Network Address Translator (*NAT*) [5] is used instead.

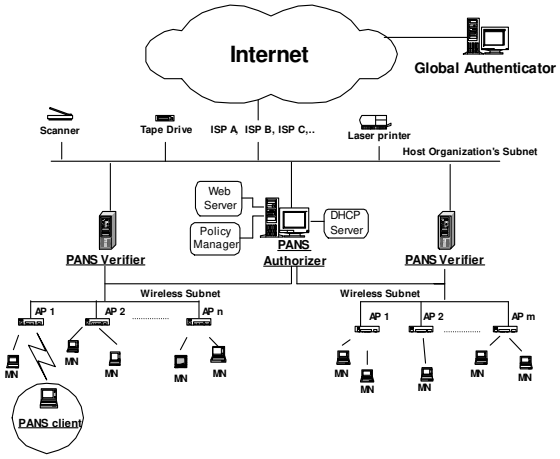

Figure 1: The various components of the CHOICE network as deployed at the Crossroad Mall in Bellevue, Washington

The IP address is given out even before the user has been authenticated to allow her access to local portals and to allow her an opportunity to download the network access software if she hasn't already done so.

The web server is the user's entry point into the CHOICE network. The local network web server is based on Active Server Pages (*ASP*) [6] and guides the user through the authentication process.

3.1.2 Authentication Database

Ideally, the CHOICE network authenticates a user by requesting from her a signed certificate containing her credentials. Thus, the CHOICE network can directly confirm the user's identity, assign the appropriate service policies for the session, and connect the user to the network seamlessly without asking for a password.

Unfortunately, there are a couple of problems that need be resolved before the system can use personal certificates for user authentication. First, it requires every user to register with a certificate authority. Since our goal is to deploy and offer public wireless network services today, we have to consider an alternative approach that is generic and readily accessible. Second, it is difficult to revoke a certificate and modify user credentials in a timely fashion. In this case, the CHOICE network may not assign the correct service policies based on the latest set of credentials.

Our current solution to this problem is to use a global authentication service that is accessible throughout the Internet. Our system uses MS Passport [8] as the authentication database. Several factors motivated our choice of MS Passport. First, its wide availability enables us to offer network service to a substantial number of users. Second, all transactions with Passport are web-based thereby greatly enhancing the usability of the system for the layperson. Third, all transactions with Passport are carried out over *SSL* [9]. Thus there is an end-to-end secure channel between the user and the authentication service. Even if CHOICE were to be set up by an un-trusted third party, this party is not able to decrypt the user's name and password while it is being supplied to Passport.

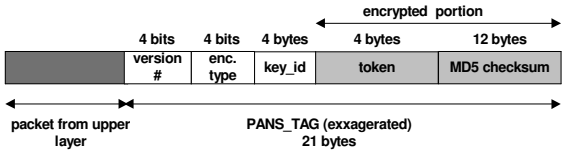

Figure 2: A key-tagged Packet. The version number, encryption type and `key_id` form the unencrypted portion, while the token and MD5 checksum are encrypted using the encryption algorithm specified under the encryption type.

3.1.3 The PANS Authorizer

The *PANS Authorizer* is a gateway to the global authentication server. To prevent unauthorized access to the Internet, the Authorizer performs IP-level filtering based on the destination IP-address of each packet. Any packet with a destination address other than the DHCP server, the WINS server, the DNS server, the local web server or the Passport server is dropped. Additionally, the Authorizer authorizes clients' access to the network upon successful completion of user authentication. It handles the task of determining service policies, generating per-user session keys, and distributing the keys to the clients and to the service gateways (i.e. *PANS Verifiers,* to be discussed next).

Each session key is valid for a finite amount of time. Depending on the host organization's preference, the key can either be automatically renewed or the user can be forced to explicitly obtain a new key when the present one is about to expire.

Once the user has been authenticated, all her communication is directed to the assigned service gateway. Individual packets are key-tagged (see Figure 2) by the client and verified by the service gateways to ensure that only authorized traffic is allowed to gain access to the Internet.

3.1.4 The PANS Verifier

The PANS Verifier handles the tasks related to per-packet verification, accounting and policy enforcement on packet transmissions between the mobile users and the public network. The PANS Client uses the Verifier as a service gateway for Internet access. The Verifier checks each packet for a valid tag generated by the client's session key. In addition, the Verifier keeps an account of the number of packets per user it has serviced and enforces policies such as QoS service-level by dropping packets from a user who violates her service agreement.

Because the task of the Authorizer and the Verifier are separated, multiple Verifiers may be deployed to

handle large volumes of traffic flow within a subnet of wireless access points. Additionally, Verifiers may be replicated to support roaming between different subnets.

3.1.5 The PANS Client

The PANS Client resides on the mobile user's device. Once the Authorizer has granted access and downloaded the key to the client machine, the PANS Client tags every outgoing packet before sending it to the Verifier (see Figure 2). Depending on the level of service the user has opted for (which may be pre-configured into the machine or arranged dynamically), the PANS client can optionally encrypt the entire packet or only a portion of the outgoing packet. The Verifier can then decrypt the packet, and remove the tag before forwarding it on to the network.

The PANS client tags packets only when the public network service is present. The client host may use the same key when it migrates to a different subnet but must negotiate a new key when it migrates to a different public network.

3.1.6 Performance

The task of per-packet verification by decrypting a packet and checking for a valid signature puts a limit on the number of connections the PANS Verifier can handle. To determine this limit we ran several tests that measured the network throughput, CPU utilization, and packet round trip time (RTT) with PANS enabled. A detailed description of the experimental methodology and analysis of the results is provided in [3].

We flooded a Verifier with PANS packets via a 100 Mbps Ethernet link and found that for bulk transfers, the network saturates before the CPU does. With the link completely saturated we observed that on average the network throughput decreased by 10% and the CPU utilization increased by 40% in the presence of PANS processing. On flooding the network with 100,000 64KB-buffers and varying the packet size during each run, we found that the per-packet RTT difference between connections with PANS enabled and without PANS was in the order of tens of microseconds.

Overall a single Verifier, which in our experiments was a 450 MHz Pentium II Dell Precision 410 workstation with 128 MB RAM, can easily handle traffic from ten 11Mbps wireless access points (APs) with PANS enabled.

4 The Mobility Problem

While PANS provides a protocol to authenticate clients and a means to control user access privileges, it does not specify any mechanism for discovering the PANS service, or a scheme for managing the client's configuration according to the available access mode in the network. To illustrate where these problems arise, let us examine the following three sample scenarios:

1. The client host migrates between the company private network and the public network. Since the company network may not be running PANS, the client host must recognize when to enable / disable the public network protocol locally.

2. The client host migrates between different subnets of the same public network. In this case, it would be undesirable to require the user to re-authenticate herself by repeating the logon process. Instead, the client should gain access in the new subnet by using the same key obtained from the previous subnet. The client host must recognize and perform any necessary changes in the routing configuration (e.g. directing traffic to a different Verifier server) and resume network operation by using the same key.

3. The client host migrates between different public networks. The client host must distinguish this from the previous scenario and ask the user to perform the logon process in the new network. After authentication has succeeded, the client host will use a new key to communicate in the new network. However, the host should save the previous key until it expires so that it could be reused upon returning to the previous network.

All three scenarios involve a combination of changing the client host's routing table, enabling / disabling the PANS module, and managing a set of keys acquired by the client. While one can change these configurations manually when the client host is relatively immobile, it would be painful, if not impractical, to have the user reconfigure the host every time she moves to a different network.

Before we describe how we solve these problems, we outline our design goals in the following section.

5 Design Criteria

The goals that influenced the design of the auto-configuration module for the CHOICE network are summarized below:

(I) Efficiency: Since our system will most likely run on wireless, mobile devices, the system should be light-weight and efficient in terms of bandwidth, memory, processing, and power.

(II) Responsiveness: The mobile host should detect a change of environment and self-configure within seconds.

(III) Ease of Deployment: We wish to avoid any changes in the existing protocol to support our auto-configuration system. Furthermore, we wish to avoid relying on any other special protocols to handle service discovery and auto-configuration. Our system should work with any

standard network stack commonly found in all types of mobile devices and operating systems.

(IV) Hardware agnostic: Our system should not require any modification to existing hardware. Also, the system should work in both wired and wireless networks.

(V) Privacy and Security: The auto-configuration system should not compromise the security and privacy models in the original PANS protocol. Namely, the auto-configuration system should ensure that the system is configured with safe and legitimate parameters.

(VI) Flexibility: We wish to examine whether employing a particular scheme would allow us to expand and implement additional features on top of PANS.

6 PANS Auto-Configuration Module

In this section we describe the requirements, design criteria, architecture, algorithms and the implementation details behind the mobility support module we have built for the CHOICE network.

6.1 Required Functionalities

The auto-configuration module needs to perform four basic functions to manage mobility between public and private networks: service discovery, bootstrapping, protocol configuration, and key management. We discuss each of these in detail below.

6.1.1 Discovery of the Public Network Service

To correctly configure the mobile host for public or private network accesses, the mobile host must first discover if a public network service is offered in the local network. To discover such a service, either a solicitation or beaconing technique may be used. We have chosen the beaconing technique for the following reasons:

- Beaconing is unidirectional so it cuts transmission overhead by half for the client host when compared to a bi-directional polling-response scheme.
- Beaconing consumes only one unit of transmission time per broadcast period, which is significantly less than the $2n$ units of transmission time consumed by n different clients in a polling-response scheme. In a wireless medium, a beaconing scheme reduces the airtime overhead and the level of traffic contention in the system.
- Polling drains more power not only from the wireless host that is broadcasting the probing messages, but also power from third-party wireless devices that must expend energy for receiving these messages broadcasted within the vicinity.
- Polling may introduce unwanted solicitation messages in private networks as client hosts continually probe for the public network service. Alternatively, a client can limit the number of broadcast queries by sending probes only when there is a good hint that the client may have migrated to another network (e.g. the hardware detects a link state change or when the host detects excessive amount of packet loss [10]). Unfortunately, such hints given by other network layers are often implementation dependent and consequently, unreliable.

6.1.2 Bootstrapping

Once a public network service is discovered via broadcasted beacons, the auto-configuration module should ensure that the mobile host has a valid IP address. (The ability to receive beacon broadcasts does *not* imply that the client has a valid IP address.) Then, the auto-configuration module sets the default gateway to the advertised Authorizer IP address, points the client's default web browser to the advertised URL of the local portal containing the authentication script, and prompts the user to begin the web-based authentication process.

6.1.3 Protocol Configuration

The auto-configuration module controls when the local PANS protocol driver should start or stop tagging and possibly encrypting/decrypting packets at the mobile host. This depends on whether the mobile host is inside a public or a private network. When the mobile host is in a public network, the default gateway must be set to the public network's Authorizer or Verifier, depending whether the user has been authenticated and obtained a valid session key. When the mobile host roams from one subnet to another within the public network, the mobile host must detect the migration and set its default gateway to a Verifier in the new subnet. When the mobile host leaves the public network, the local PANS driver should stop tagging packets and the default gateway should be set to the default system values (e.g. DHCP).

6.1.4 Key Management

After the user has been authenticated in the public network, she is given a session key for accessing the Internet. The session key expires after a pre-defined period. Over time, a user can enter and exit different public networks and collect a set of session keys. Therefore, whenever the user enters a public network, the auto-configuration system should determine whether the user currently has a valid key. If so, the system should automatically bypass the authentication procedure and pass the correct key to the local PANS protocol driver to access the public network. When the key is about to expire, the auto-configuration system should initiate a procedure for renewing the session key.

6.2 Architecture and Implementation

Due to the considerations listed above, we have designed and built an auto-configuration system that uses beacons to discover the PANS service and obtain the necessary configuration parameters to bootstrap the authentication process. When the client migrates out of the

public network service, it no longer receives any beacons. The client times out and resumes normal networking operation by disabling the special mode at the local PANS driver and resetting the default gateway value.

Our scheme is very similar to the broadcast advertising schemes found in Mobile IPv4 and Mobile IPv6 [11] except that it also supports a number of other extended features such as client-side key management, a system-wide fail-over mechanism, and location-sensitive messaging. The next section describes the components and the algorithm of the system.

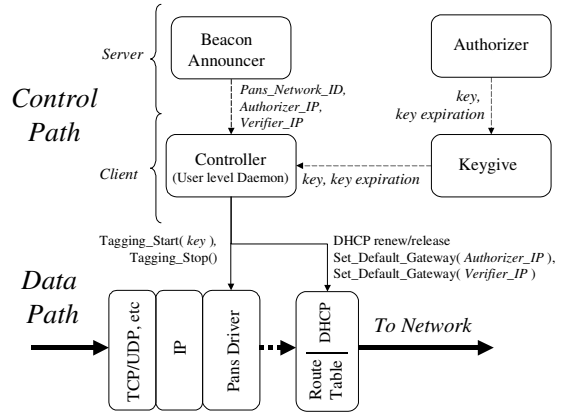

Figure 3 Architecture of the auto-configuration system for PANS

6.2.1 Components

Figure 3 illustrates the architecture for supporting auto-configuration in PANS. The diagram divides two types of data flow. The control path illustrates the flow of configuration parameters and control signals of the auto-configuration system. Sitting below the control path is the data path, which illustrates how data packets flow through the network stack incorporated with an intermediate PANS protocol driver for packet tagging.

Inside the control path are two types of modules. The server modules, which consist of the *Beacon Announcer* and the *Authorizer*, sit inside the Authorizer server and sends configuration parameters and session keys to the client modules. The Beacon Announcer periodically broadcasts a beacon, which contains a unique PANS network_id, a subnet_mask, a URL of the user log-on web page, the current Authorizer_ip and Verifier_ip addresses.

As mentioned in Section 4, the PANS Authorizer serves as a proxy for a global authentication server such as MS Passport. After the user completes the authentication process, the Authorizer establishes a secure *SSL* connection with the client's web browser. Using this connection, the Authorizer delivers the session key and key expiration values to the key manager inside the client's Controller daemon.

Although using the web browser's *SSL* service saves us from implementing a special secure protocol for key delivery, we now need a way to direct the session key from the web browser to the key manager. To do this, the ASP script on the Authorizer delivers the key values via a MIME-typed data stream, which triggers the web browser to launch the registered *Keygive* user level program[1]. The web browser then pipes the key values to the Keygive module, which in turn hands them over to the Controller.

Notice that the Verifier_ip values are deliberately transmitted inside the broadcast beacon instead of being transmitted alongside with the session key. This is done to allow those clients who have migrated to a different subnet but still hold a valid session key to directly access the local Verifier without repeating a full authentication process. Furthermore, such a design supports a useful system-wide fail-over feature and load-balancing mechanisms. For example, the *Beacon Announcer* can broadcast a different Verifier_ip to instantly migrate all the clients to use a backup gateway.

The heart of the auto-configuration system is the Controller, which runs a finite state machine to handle the external events generated by the server modules and to coordinate the tasks of discovering a public network service, bootstrapping the authentication process, configuring the PANS protocol driver and the routing table, and managing session keys on the client host.

The Controller listens to two well-known ports: one detects beacons coming from the Announcer, and the other receives the session key and expiration values from the Keygive module. The Controller stores the session key value into a table indexed by the network_id associated with each key. The Controller implements an earliest-expiry-time replacement policy and invalidates a session key entry whenever it expires. Then by matching a valid row entry with the currently advertised network_id, the Controller can use the appropriate session key to configure the local PANS driver via an *ioctl* call.

6.2.2 Operation

Figure 4 depicts the Controller's finite state machine. Bootstrapping starts when the Controller detects the first beacon. The Controller uses the network_id and the subnet_mask in the beacon to distinguish whether the client has roamed to a different subnet within the same network or migrated to a different public network. In either case, the Controller verifies that client has a valid IP address to operate in the new subnet and updates the

[1] By accepting keys via MIME-typed data streams, the Keygive program may be launched by malicious web portals. Although not currently implemented, we can easily extend the Keygive program to authenticate the key delivery channel via certificates or other secure mechanism such as S/MIME [25].

address if necessary. Our system currently relies on DHCP to obtain a dynamic address assignment and set the default DNS server. To ensure timely address assignment, the Controller will force a DHCP request and loops in the Detect state until the client host receives a valid address that is contained in the subnet advertised by the beacon. This is done to handle roaming problems where DHCP fails to recognize a network migration due to inconsistent media sensing implementations.

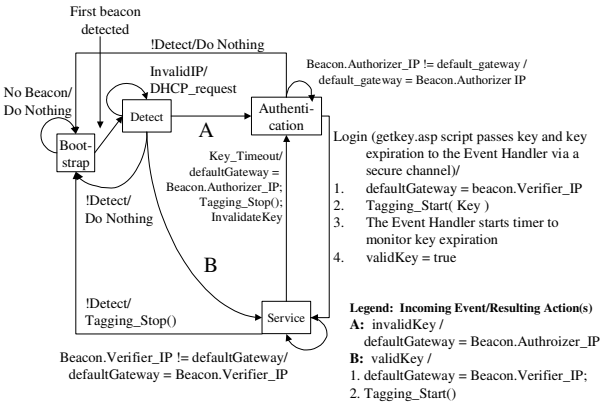

Figure 4 Controller Finite State Machine

Next, the Controller checks the key management table to determine if the client currently possesses a valid key for the current network by checking the beacon's network_id. If so, it bypasses the login and authentication procedures and sets the default gateway to the advertised Verifier. The Controller enters the Service state so that the client can seamlessly resume the previous PANS session. This setup supports client migration between subnets as well as migration between different public networks.

If the client does not have a valid key, the client has entered a new public service and must begin the authentication process to gain full access to the network. The Controller enters the Authentication state where it sets the client's default gateway to the advertised Authorizer IP address and extracts from the beacon the URL of the authentication web page. Then, the Controller pops up a greeting message to notify the user about the discovery of a new public network service and displays the URL. If the user wishes to join the service, the user launches her default web browser to the given URL, and begins the web-based authentications process. The Controller waits in the Authentication state until the authentication succeeds or until the client migrates out of the current network. When the authentication succeeds, the Controller enters the Service state by saving the session key into the key management table, setting the default gateway to the Verifier, passing the session key to the PANS driver, and enabling packet tagging via an *ioctl*.

The user gains full access to the Internet in the Service state. Service can be interrupted when the user migrates to another network, when the session key expires, or when no beacons are detected. In the case of user migration, we repeat the bootstrapping process. When the key of the ongoing session expires, the Controller negotiates for a new session key. The user may be prompted for a confirmation or the renewal process can happen automatically, depending on the user's preference. When no beacons are detected for a fixed period, the public network service is no longer available. The Controller disables the PANS driver and configures the client to the system's default parameters for accessing private networks.

As a final remark, we wish to emphasize how we have decoupled key management and mobility management so that the network access protocol (PANS) and the auto-configuration mechanism can work independently of each other. Our system includes all the network parameters within the beacon to support the bootstrapping process. Thus a client possessing a valid session key can immediately access the network without repeating the authentication process. Likewise, the Controller is free to refresh a client's key during an authenticated session without affecting the client host's network configuration. In Section 7, we will explain how this decoupling of key and mobility management also helps us implement a scalable key distribution scheme as well as fail-over and load balancing mechanisms for the public network infrastructure.

7 Beyond Mobility - Extending the System

One of the main goals of the CHOICE Network project is to deploy public network access service in large settings such as major conference centers, airports, shopping malls, and the like. Thus, the network access service must itself be scalable and highly available.

We have considered these issues when designing the auto-configuration mechanism for PANS, and have found ways to extend the beaconing mechanism to help the network access service attain scalability and fault-tolerance. We have found other applications for the beaconing mechanism as well. We will describe the various extensions we have considered in the next few subsections.

7.1 Auto-Configuration to Provide High Availability and Scalability

Because the Authorizer and Verifier contain the set of active keys in the network, the public network service must provide a fail-over mechanism to handle the case when a Verifier fails. To prevent loss of information, a

service provider may install multiple backup Verifiers that replicate the table of keys currently active in the network.

In addition, service providers may scale their service by installing multiple Verifiers to share high traffic loads.

Hence, the requirement for achieving high availability and scalability is to configure each client to use the appropriate gateway when accessing the network.

We extend our auto-configuration system to support fail-over and load balancing as follows. For each beacon, we advertise a *vector* of Verifier_ip addresses instead of one Verifier_ip. The vector represents the set of operational Verifiers, which excludes the set of backup verifiers. The first element of the vector is the "preferred" gateway of which new clients should use when they successfully log on to the network for the first time. Thus, to balance the traffic load among the operational Verifiers, the advertised value for the first element is rotated according to the load of the network.

When a gateway fails, the Authorizer advertises the backup server as the last element of the vector and removes the IP address of the failed gateway from the vector. Thus, clients in the network continually scan the vector in every beacon to ensure that their gateway is available. If not, the client simply switches to the backup Verifier that is advertised as the last element in the vector. Because the backup Verifier contains a copy of all active keys in the Verifier that failed, the transition should occur smoothly, with minimal disruptions, if any, to all ongoing network transaction.

7.2 Subnetting and Scalable Key Distribution

For a number of administrative reasons, subnetting may be required to scale large public networks. Each subnet has its own address space and its own set of Authorizer and Verifier gateways. When a mobile client roams across subnets, the host must change its IP address and set its default gateway to the Verifier advertised in the new subnet. In Section 6, we have already discussed how our dynamic host auto-configuration system supports mobility for the client host. However, we have glossed over the issue about how the keys are to be distributed behind the network.

One simple solution is to distribute keys globally within the network infrastructure. However, this approach clearly does not scale well as the number of users grow in the network.

The requirement for scalable key distribution in the network infrastructure is to avoid sharing key information globally among all the Verifiers in the network. Each Verifier should be responsible for managing the set of keys belonging to the set of active clients in its own subnet. Under this requirement, the network must be able to migrate keys to the Verifier that handles traffic from the client's current location.

Here, the auto-configuration system can help. The Controller can keep a history of the subnet that the client has previously visited. Then when the client roams to another subnet, the Controller can automatically request a key migration from the Authorizer server in the new subnet. The request contains an encrypted portion containing the client's token and an unencrypted portion, which includes the client's key identifier and the address of the original Authorizer that issued the key. The Authorizer in the new subnet would forward the request to the original Authorizer, which authenticates the request by checking the encrypted token against the token stored in its table. If the verification succeeds, the original Authorizer will return the requested key to the new Authorizer via secure channels. After the new Authorizer receives the key, it redistributes the key among the Verifiers in the new subnet and acknowledges the client. Thus, the client migrates to a new subnet seamlessly by using this scalable, on-demand approach to key distribution.

7.3 Location Services

In a wireless network, the beaconing mechanism could also be extended to provide certain coarse-grain location information. We will outline two applications below.

7.3.1 Network Usage Service Metric

A very practical piece of information to include in the beacon is a metric that represents the network's load. For example, as the Verifier server becomes highly loaded, the Authorizer can advertise a low service quality metric to the clients. The Controller can be modified to interpret these metrics and notify the user about the current conditions of the network. Hence, users can change their expectations or access behavior according to the system's feedback. For example, if there is a cost associated with accessing the public network, then a user can decide whether it is worth the cost to register with a public network that is presently congested.

Finding an appropriate metric for this purpose is still an open problem. It is unlikely that the Authorizer or Verifier server would ever become the bottleneck of the system (provided that the administrator has scaled the system using the suggested techniques). Rather, the individual access points in the wireless network are more likely to become the bottlenecks. Hence, we can extract load information from each access point and redistribute this "metric" to the associated clients. To our knowledge, the availability of such load information varies between different access point implementations. It would convenient if the API for extracting such information were standardized.

7.3.2 Coarse level Location Information

Generally, device drivers of wireless network interfaces (WNIC) can obtain the id of the Access Point (AP) with which the WNIC is associated. If so, a client can download a map of all APs within the vicinity and use the id to locate its position on the map. A client can then infer that her location is roughly the same as the AP with which she is associated. Although this is a very coarse-grain approach for identifying the user's location when compared to the proposed alternatives [21][22][23], it is nevertheless a useful feature (especially in large settings such as the airport, where many APs are deployed) that can be readily implemented in our system.

Another simple but useful location-sensitive application is "coded messaging." Instead of mapping the access point identifier to a physical coordinate on a map, the Controller can map the identifier to a table of messages. Thus, depending on the user's preference, the Controller can pop up messages to notify the user about a special event that is happening near the access point of which the client is associated (e.g. notification of a special promotion at a nearby coffee shop). The table can also include a time-index so that messages can pop up at specific times during the day.

Finally, we would like to extend a word of caution. The purpose of this section is to illustrate the power behind a beaconing system and to illustrate how it could be used to build simple but useful services. It is not well-suited for implementing heavy-duty service discovery applications mentioned in [19][20]. In particular, we must be careful not to overload the beacon with too much data as our design goal is to keep the auto-configuration system lightweight. The examples above show how to do this by means of mapping compact codes contained in the beacon with a downloadable table containing the full information required for the application.

8 Discussions

In this section, we will discuss some issues that need to be considered for providing secure and seamless mobility support in our auto-configuration system.

8.1 Mobile IP vs. Auto-Configuration

We wish to emphasize that the set of mobility problems addressed by our auto-configuration system is different from those addressed by Mobile IP and other similar IP-level migration protocols. Mobile IP is primarily concerned with locating the mobile host and re-routing packets to the host's current destination. In contrast, our protocol is concerned with configuring the host to migrate between public and private networks.

Nevertheless, both systems do share some similarities. When the mobile host migrates to a foreign network, the protocol employs a similar beaconing strategy to probe for a Foreign Agent and configure the local Mobile IP stack to the correct mode of operation. Despite this similarity, we chose not to extend Mobile IP to support the auto-configuration requirements in PANS. Our goal is to be protocol agnostic so we avoided tying our system to any specific protocols. Hence, any protocol, including Mobile IP, will continue to operate seamlessly on our system.

8.2 Low-Level Configuration

There is one situation that may prevent our auto-configuration system from migrating a client between public and private networks. The problem is caused by special configurations in the WNIC. As mentioned in the introductory sections, some private networks use the wired equivalency protocol (WEP) to secure the wireless link [2]. A user must manually enable the WEP key settings in the WNIC driver, otherwise none of the beacon packets (IP-level broadcast packets) would reach the client host.

We are aware of some on-going efforts that specifically address this issue. For example, future mobile clients will automatically cycle through several different pre-configured WEP keys in an attempt to associate with the wireless network. When all keys fail, they will try to associate with the network with the WEP key disabled. With the appropriate extensions and API for supporting mobility in this manner, our system should migrate clients seamlessly between all types of public and private networks.

8.3 High-Level Configuration

Problems with application settings may arise as the user migrates between networks. For example, a client's web browser may default to a proxy server in the corporate network. After migrating to the public network, the user might find excessive browser delays caused by timeouts as the browser tries to locate the default proxy. To prevent such problems, the applications should be made aware of the host's mobility [25].

We should mention that another solution to this problem is to employ Mobile IP. However, using such techniques may induce certain limitations [24]. For example, Mobile IP, in certain cases, may tunnel packets destined for the mobile agent from the Home Agent. If the home agent is situated in the corporate network, the client traffic will be governed by the policies imposed by the corporate proxy. In contrast, the user may gain full access to the Internet via other end-to-end mobility mechanisms [25] that allow applications to open direct connections and assume the access policies defined by the public network.

8.4 Beaconing, Polling and Issues about Media Sensing in Wireless Networks

Before auto-configuration can be triggered, the client host must implement a mechanism to detect when it has migrated to another subnet or to another network. We solve this problem by comparing the network_id values between beacons. This is similar to the mobility detection algorithms proposed by Mobile IP [11].

Another common solution to the mobility detection problem is to rely on a link-level (a media sensing) mechanism to trigger the auto-configuration mechanism when there is a change in the client's link state. This scheme works well for DHCP in wired networks, which, upon a link-state change, broadcasts a configuration request message to retrieve a dynamic address assignment. In most instances, DHCP is able to reconfigure the client without the use of beaconing or the extended use of polling; the polling stops as soon as the DHCP server responds to the client's message. There is no need to rely on polling or beaconing to detect migration because the next link-state change would trigger DHCP to reconfigure the client.

While the media sensing method works well for DHCP, it does not provide adequate micro-mobility detection in wireless networks. Consider when a client roams between two overlapping APs belonging to two different subnets. From our experience, some of the WNICs we have experimented with do not trigger DHCP to verify its client address and reconfigure the client when it is necessary. From the WNIC's point of view, this is the correct behavior because it is agnostic to the IP-level topology. The WNIC's default behavior is to handle the common case where the client stays within the same subnet as she roams between two APs.

The absence of a set of well-defined, consistent media sensing capabilities across different network interface technologies and their implementations has reinforced our design decision to use beacons, which is a hardware agnostic approach for mobility detection. We should note that the tradeoff of increasing beaconing frequency for reducing mobility detection time should not be significant. For instance, we can send beacons (on the order of a few hundred bytes) at the rate of 1Hz, which translates to negligible overhead in an 11Mbps wireless network.

8.5 Security

A good question to ask when examining the design of any auto-configuration system is how well does the system enforce security in the face of malicious attacks. In this section, we will concentrate on security issues that affect the auto-configuration part of the PANS system. For a full discussion about security topics concerning the PANS protocol, please refer to [3]. We will assume these security features from the PANS protocol throughout our discussion:

- The key and any other relevant parameters can be downloaded securely from the Authorizer to the client during the authentication process.
- The client can use the full packet encryption feature provided in PANS to increase security.

The auto-configuration system uses beacons to trigger client host configuration. Hence, the beacon becomes the entry point for possible attacks against the auto-configuration mechanism. Below, we will illustrate two types of attacks against our system and suggest possible security measures to guard against them.

8.5.1 Denial of Service (DoS)

A malicious user may learn the beaconing frequency and jam or intercepts the beacons at the predicted rate. Without detecting the beacons, the clients are denied access to the public network.

While we cannot prevent all forms of DoS attacks (such as jamming the entire wireless channel), we should make the attack difficult and/or detectable so that the service provider is alert to such an attack. First, the beaconing intervals can be randomized so that the attacker must either try to jam the entire channel (in the wireless network) or intercept the beacons on the physical network. These measures increase the difficulty of the attack by increasing the attacker's exposure to the system, thus reducing the chances of that attack go unnoticed. Whether there is an attack or not, the system should implement a network monitoring mechanism to ensure that the public network is operating normally. As an example, receivers of the network monitoring system can be installed throughout the physical area of a wireless public network. These receivers will monitor the frequency and integrity of each beacon being broadcasted by the individual APs. Although a malicious attacker can fool a receiver by replaying short-range beacons towards it, the attacker must devise such a device and possibly leave traces of evidence about the attack.

8.5.2 Hijacking

An attacker can redirect a client's packet stream by sending a false beacon containing an illegitimate Authorizer and/or Verifier address. The client can guard against this by performing integrity checks and authentication for each beacon. However, such technique is very costly and should be avoided. As an alternative, the network can set up a pair of public and private keys. In this scheme, the client must authenticate the Authorizer upon connection by, for example, checking its certificate. Then the client obtains the public key from the trusted Authorizer after she successfully gains access to the network. Just as she migrates her connections to a Verifier server, the client will issue a challenge to the Verifier.

The Verifier must return the challenge encrypted with the network's private key. The client will authenticate the Verifier by decrypting the challenge with the network's public key to see if it matches the original challenge it had sent to the Verifier. As an added measure of security, the client should use full packet encryption as provided by the PANS protocol.

9 Related Work

We are aware of a considerable amount of on-going work in the areas of Internet protocol design that addresses pieces of functionality that the CHOICE network provides. Although CHOICE combines and covers a broad range of ideas in existing work, we will discuss the work most relevant to our authenticated network access system and to our dynamic host auto-configuration system. We point the interested reader to [3] where additional details and comparisons are provided.

In the area of providing authenticated access to users, the two layer-2 mechanisms described in the IEEE 802.11 standard [2] (a) MAC-level filtering, and (b) the wired equivalency protocol (WEP) are insufficient for deployment in a public wireless networks. MAC-level filtering is difficult to manage and doesn't scale well, and WEP lacks the necessary hardware support for large-scale key management on a per-user basis. Other hardware-centric proposals include [12], [30], and [14]. Of these the most recent and promising one is the IEEE 801.1X standards committee's port-based network access control proposal which carries out layer-2 authentication by carrying the Extensible Authentication Protocol (EAP) frame within the Ethernet frame [12]. However, all of these proposals address only one aspect in our system and they do not consider issues like accounting, service quality, and user mobility. This last point is particularly important and has been discussed in detail in this paper.

The only fully deployed and documented authenticated network access system that we are aware of is the SPINACH system developed as part of the MosquitoNet project at Stanford University [13]. The strengths of SPINACH are the innovative reuse of existing infrastructure with no requirement for additional software in the client. However, this advantage also limits its functionality to user authentication only. The CHOICE system requires client side software but because of this is able to incorporate service quality and mobility support in addition to authentication, privacy and security. Also, without IPsec [7] in place, the SPINACH system does not protect against hardware spoofing, whereas our system does.

As mentioned, there are some Internet protocols that can be combined to build part of our system. For example, IPsec authentication header (AH) [15], IPsec encapsulating security payload (ESP) [16] and IKE [17] can be used to solve the problem of privacy and security. However, the strength, power, and feature-richness of these protocols come at the cost of overhead that may be slightly too expensive for the average handheld wireless device. In CHOICE, we reduce the cost of bearing last-hop encryption by implementing a lightweight protocol to meet the specific needs of our service model. Where the need arises, clients can still use IPsec on CHOICE for strong protection of their individual end-to-end connections.

Moreover, IPsec couples user keys and security association tightly with IP level information. This directly impacts our goal of supporting roaming users whose IP address changes frequently. This point has been addressed in this paper where we have described a system that decouples key information from IP level information and consequently supports mobility with fast hand-offs.

In the area of supporting mobility, there is Mobile-IP (v4 and v6) [11], which employs a service discovery scheme based on ICMP router discovery. The method of service discovery is similar to ours except that our system does not provide a mechanism to probe the network for the target service. Service discovery protocols, such as Berkeley SDS [20] and MIT INS [19] can be used for locating and using some network services. However, these systems mainly address the problem of handling a large number of services in a highly dynamic environment, which is overkill for our application.

In the area of host configuration, DHCP is perhaps the most relevant piece of work [4]. Our system relies on it to configure the client's IP address. Although DHCP provides a set of configurable options field, we have defined a separate beaconing mechanism for our host configuration application. The primary reason for using a beaconing mechanism is to support fast mobility detection, dynamic failure recovery, and location information delivery.

CHOICE is designed with a specific set of user-centric requirements and tries to combine the strengths and features of the on-going efforts mentioned in this section to build a comprehensive system that is self-contained, hardware agnostic, and protocol agnostic. We have designed it so that the client software can be downloaded and installed on-site giving the service provider considerable flexibility in personalization.

10 Conclusion

The CHOICE network is a case study of computing and communications in public places. We have designed and deployed this network at a popular mall with the hope that it will provide us a research platform for studying how the general public actually uses such networks and the sorts of services they care about. We are un-

aware of any working, deployed and documented system that addresses all the issues we tackle in our network. In this paper we focus on the specific problem of managing nomadic users as they move between differently configured public and private networks. That this problem is real is confirmed by our experience in supporting corporate employees who have their own private wireless network. Our solution to the problem has many advantages. Specifically, (a) It supports dynamic configuration of client devices, without user intervention, as nomadic users roam between public and private networks. (b) It achieves high availability of network services, network scaling and load-balancing, and (c) it supports location services that are currently not available in other networks. In describing our solutions we make the case that achieving true device mobility without any user intervention requires that we resolve many issues beyond the ones being worked on within standards committees like the IETF and the IEEE. The existence of standards in device programming and access point programming can help us achieve our ultimate of seamless mobility.

Acknowledgement

We would like to acknowledge and thank several individuals who have helped develop the CHOICE network. In particular, Anand Balachandran and Srinivasan Venkatachary are two of the original designers and implementers of PANS. Stephen Dahl helped us deploy the network at the Crossroads Mall; Pierre De Vries handled the legal formalities and helped us with usability issues while being our liaison with the product groups. Paul Hoeffer designed our web interaction. We also thank Dave Andersen of MIT, Prof. Dave Johnson of Rice University, and Prof. Mary Baker of Stanford University for the well appreciated constructive discussions.

References

[1] ITU-R Rec. M. 1225, "Guidelines for Evaluation of Radio Transmission Technologies for IMT-2000," 1999.

[2] IEEE 802.11b/D3.0, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification: High Speed Physical Layer (PHY) Extensions in the 2.4 GHz Band, " 1999.

[3] P. Bahl, A. Balachandran, and S. Venkatachary, "The CHOICE Network – Broadband Wireless Internet Access in Public Places," MSR-TR-2000, February 2000

[4] R. Droms, "Dynamic Host Configuration Protocol," *IETF RFC 2131*, March 1997, http://www.ietf.org/rfc/ rfc2131.txt

[5] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. de. Groot, "Address Allocation for Private Internets," *IETF RFC 1597*, March 1994, http://www.ietf.org/rfc/rfc1597.txt

[6] Active Server Pages: http://msdn.microsoft.com/ workshop/server/asp/ASPover.asp

[7] R. Atkinson, "Security Architecture for the Internet Protocol", *IETF RFC 2401*, November 1998, http://www.ietf.org/rfc/ rfc2401.txt

[8] MS Passport: http:/www.passport.com

[9] T. Elgamal, S. Cotter, and the Netscape Security Team, "Netscape Security: Open-standard Solutions for the Enterprise,1998",http://developer.netscape.com/docs/manuals/ security/scwp

[10] R. Braden, "Requirements for Internet Hosts Communication Layers, *IETF RFC 1122*, October 1989

[11] Internet drafts from the IETF Working Group, "IP Routing for Mobile and Wireless Hosts (Mobile IP)," http://www.ietf.org/html.charters/mobileip-charter.html

[12] *IEEE Draft P802.1x/D1,* "Port Based Network Access Control," September 1999

[13] G. Appenzeller, M. Roussopoulos, and M. Baker, "User-Friendly Access Control for Public Network Ports," *Proceedings of INFOCOM '99*, March 1999

[14] E. A. Napjus, "NetBar - Carnegie Mellon's Solution to Authenticated Access for Mobile Machines," CMU White Paper, http://www.net.cmu.edu/docs/arch/netbar.html

[15] S. Kent and R. Atkinson, "IP Authentication Header," *IETF RFC 2402,* Nov. 1998, http://www.ietf.org/rfc/ rfc2402.txt

[16] S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)," *IETF RFC 2406,* November 1998, http://www.ietf.org/rfc/ rfc2406.txt

[17] D. Harkins, and D. Carrel, "The Internet Key Exchange (IKE)," *IETF RFC 2409*, November 1998, http://www.ietf.org/rfc/ rfc2409.txt

[18] Microsoft Virtual Private Networking (VPN) White Paper, http://www.microsoft.com/ntserver/commserv/ deployment/planguides/VPNSecurity.asp

[19] W. Adjie-Winoto, W., E. Schwartz, H. Balakrishnan,, and J. Lilley, "The Design and Implementation of an Intentional Naming System.," In *Proceedings ACM Symposium on Operating Systems Principles* (Kiawah Island, SC, Dec. 1999), pp. 186-201.

[20] S. Czerwinski,, B. Zhao, T. Hodes, A. Joseph, and R. Katz, "An Architecture for a Secure Service Discovery Service," In *Proceedings of the ACM/IEEE MOBICOM* (Seattle, WA, Aug. 1999), 24-35

[21] Bahl, P., and Padmanabhan, V., "RADAR: An In Building RF-based User Location and Tracking System." In *Proc. IEEE INFOCOM* (Tel-Aviv, Israel, Mar. 2000).

[22] Want, R., Hopper, A., Falcao,V., and Gibbons, J., "The Active Badge Location System. ACM Trnsactions on Information Systems 101," (January 1992), 91-102

[23] Priyanth, N., Chakraborty, A., Balakrishnan, B., "The Cricket Location-Support System," In *Proc. ACM/IEEE MOBICOM 2000* (Boston, MA, Aug. 2000).

[24] Cheshire, S., Baker, M., "Internet Mobility 4x4." In *Proc. SIGCOMM 1996*, August 1996.

[25] Snoeren, A., Balakrishnan, B., "An End-to-End Approach to Host Mobility," In *Proc. ACM/IEEE MOBICOM 2000* (Boston, MA, Aug. 2000).

[26] Ramsdell, B., "S/MIME Version 3 Message Specification," IETF RFC 2633, June 1999, http://www.ietf.org/rfc/rfc2633.txt

[27] Hodes, T. D., Katz, R. H., "Composable Ad-hoc Location-based Services for Heterogeneous Mobile Clients," In *Proc. ACM/IEEE MOBICOM 1997* (Budapest, Hungary, Sept. 1997).

[28] Loat, C., Gross, G., Gommons, L., Vollbrecht, J., Spence, D., "Generic AAA Architecture," *IETF RFC 2903*, August, 2000.

[29] Schneirer, B., "Applied Cryptography," John Wiley & Sons, Inc., 1996.

[30] D. L. Wasley, "Authenticating Aperiodic Connections to the Campus Network," June 1996, http://www.ucop.edu/ irc/wp/ wp_Reports/wpr005/wpr005_Wasley.html