# The CHOICE Network: Dynamic Host Configuration for Managing Mobility between Public and Private Networks

Allen Miu
Victor Bahl

August 2000

Technical Report
MSR-TR-2000-85

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

## Acknowledgements

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 Prof. Dave Johnson of Rice University, and Prof. Mary Baker of Stanford University for the well appreciated constructive discussions.

# Dynamic Host Configuration for Managing Mobility between Public and Private Networks

**Allen Miu**
Massachusetts Institute of Technology
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 and home) network and consequently, the two networks are often configured differently. The existence of such networks motivates our need to improve support and management of nomadic users who frequently roam between them. Using a real-world public network as a case study, we discuss this problem in detail and describe a solution that allows client devices to dynamically configure themselves 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 there is a greater demand for Internet connectivity than anytime before in the history of the Internet. Our nations economies and businesses rely heavily on people having this 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 wireless network, called the *CHOICE* network, 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 *25X* to *50X* greater than 2.5G and 3G speeds [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 list, 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.

Our network is deployed in a popular shopping mall, called the Crossroads Mall in Bellevue, Washington, USA (URL: http://www.mschoice.com). In deploying this network we were faced with many challenges but the one that we focus on in this paper is the problem of accommodating roaming users who frequently move between private and public networks. Concretely, the Bellevue Crossroads Mall is fairly close to the Microsoft's corporate headquarters in Redmond, Washington (approximately 3 miles). Microsoft employees routinely visit this mall's international food court during lunch hours, and hold business meetings at the resident Starbuck's coffee shop at different times of the day. These employees have access to their own corporate campus-wide wireless LAN, which is deployed in over 30 multistory buildings. Consequently, a large number of them have notebooks and PDA equipped with wireless LAN cards that they use exclusively for network connectivity. We were thus faced with the challenge of supporting these users in a way that they don't have to do anything to their device configurations as they roam between their private corporate network and the public **CRO**ssroad's **W**ireless **N**etwork (a.k.a CROWN).

Typical usage scenarios for private and public networks are different and consequently, these networks are generally configured differently. This case is confirmed in how Microsoft's internal wireless network is configured compared to the how the publicly available CROWN is configured. 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 applied to keep out unknown persons from accessing internal private networks. Public networks on the other hand 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. So 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 by-and-large corporations use some sort of a pre-configured shared key mechanism with hardware encryption to secure network access. Public networks on the other hand end-up doing 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 may not do anything; hardware encryption with a shared key allows users access through the network ports. However, when accessing the public network (special mode), the client runs through an authentication process and starts using a specialized network access protocol, which gets them different types of commercially interesting services.

With these issues in mind, we developed a mobility support mechanism that allows devices to automatically figure how to establish/re-establish network connectivity as roaming users migrated 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 describe PANS briefly 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. It is lightweight and protocol agnostic. PANS is not bound to any higher-layer Internet protocols and can be used in both WAP [25] and IP devices.
b. It is hardware agnostic. PANS does not depend on any special hardware feature other than basic network access. Consequently, it can be deployed over legacy hardware today. (For a somewhat contrasting approach to authentication see Section 8).
c. It is self-contained. It combines a lot of functionality into a compact design without being dependent on the co-existence of any upper-layer protocols. It can be downloaded on-site.

In addition to the above, which are discussed in [3], this paper described the details behind the following additional features:

d. It supports dynamic configuration of client devices, without user intervention, as nomadic users roam between public and private networks.
e. It extends the dynamic configuration mechanism for achieving high availability, scaling and load-balancing, and
f. It supports location services currently not available in other networks.

The rest of this paper is organized as follows: Section 2 sets up the stage by describing a typical scenario of user interaction with public and private networks. Section 3 then articulates the precise mobility problem that we are interested in solving. In Section 4, we describe the system components of the CHOICE network. In Section 5, we discuss our solution in dealing with the mobility problem. In Section 6, we explain how our mobility support solution can be extended for providing a highly available system that can scale well and has load balancing. We discuss on-going and future work for achieving true mobility in Section 7 and survey related work in the field in Section 8. Finally, we conclude in Section 9.

## 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 which 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 which starts to boot and as part of this boot-up process automatically generates a DHCP request [4]. The local DHCP server picks up this request and leases a short-term routable IP address to the device. Once the boot-up has completed, the user launches her web browser and points it to http://choice. The local DNS server resolves the name Choice to the address of the host organization's web server. Thus, the user only has to remember the word "choice" to get to the local web server anywhere the CHOICE network is available. At this point, if 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 to get Internet access she has to log-on and establish 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. The user's identity and password are sent encrypted via Secure Socket Layer (SSL) [9] over an *https* connection so no one accept the user and the database are privy to this information. 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 the policy the network generates a unique key and sends it to the PANS client via another SSL connection. At this point the user's web browser automatically refreshes to the local portal and Internet access is now possible. Depending on the number of services this user has subscribed to she can now get relevant messages or information about these on her notebook. For example, if she subscribes to a location-based buddy-list the system can inform her if her colleagues are in close proximity (provided they are connected to the local network as well). 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 mobility problem precisely.

## 3 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, nor 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 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 impossible, to have the user reconfigure the host every time she moves to a different network.

Before we describe how we solve these problems we need to provide some details about the CHOICE network and introduce some of the terminology. This we do in the following section.

## 4 The Choice Network

In this section, we describe the key system components of the CHOICE network. Our description is relatively brief as we refer the reader to [3] for greater details. Figure 1 illustrates the different components of the CHOICE network as has been deployed at the Crossroads Mall.

### 4.1 System Components

The CHOICE network has several system components that manage address allocation, authentication, authorization, security, accounting, and last-hop QoS. We briefly explain these in relation to our deployment.

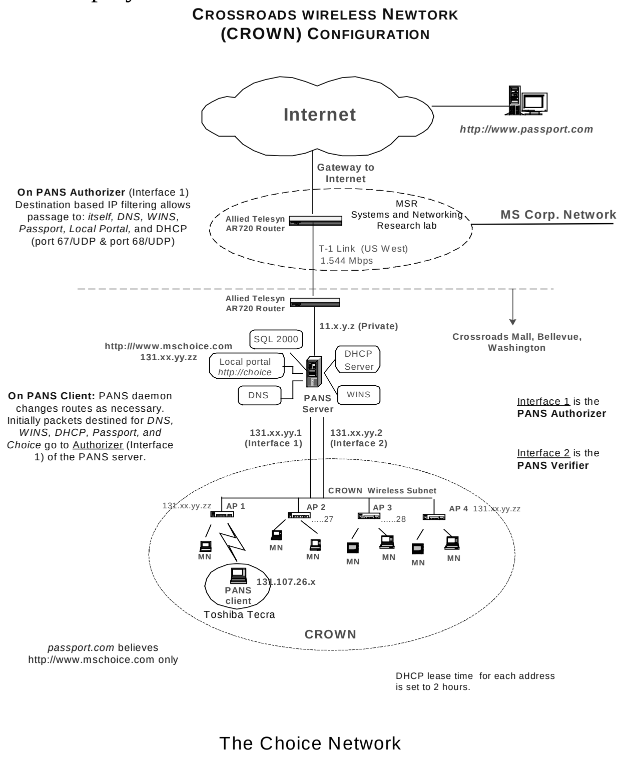

CROSSROADS WIRELESS NEWTORK
(CROWN) CONFIGURATION
Internet
http://www.passport.com
Gateway to Internet
On PANS Authorizer (Interface 1) Destination based IP filtering allows passage to: itself, DNS, WINS, Passport, Local Portal, and DHCP (port 67/UDP & port 68/UDP)
Allied Telesyn AR720 Router
MSR Systems and Networking Research lab
MS Corp. Network
T-1 Link (US West) 1.544 Mbps
Allied Telesyn AR720 Router
11.x.y.z (Private)
Crossroads Mall, Bellevue, Washington
SQL 2000
http:///www.mschoice.com 131.xx.yy.zz
Local portal http://choice
DHCP Server
DNS
WINS
PANS Server
On PANS Client: PANS daemon changes routes as necessary. Initially packets destined for DNS, WINS, DHCP, Passport, and Choice go to Authorizer (Interface 1) of the PANS server.
131.xx.yy.1 (Interface 1)
131.xx.yy.2 (Interface 2)
Interface 1 is the PANS Authorizer
Interface 2 is the PANS Verifier
CROWN Wireless Subnet
131.xx.yy.zz
AP 1
AP 2
.....27
AP 3
......28
AP 4
131.xx.yy.zz
MN
131.107.26.x
PANS client
Toshiba Tecra
CROWN
passport.com believes http://www.mschoice.com only
DHCP lease time for each address is set to 2 hours.
The Choice Network

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

#### 4.1.1 Address Management and Naming

The CHOICE network uses a standard DHCP server to lease out 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.

The IP address is given out even before the user has been authenticated to allow her access to information about the building's local services and to allow her an opportunity to download the network access software if she hasn't already done so. So a user can walk into any building, download the software and start using the network.

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. It would be evident that a prerequisite to the authentication process is the successful obtainment of a valid IP address and a connection to the web server. Since both these network connections have to go through prior to authentication, the task of the PANS server module is divided between two sub modules, which we discuss below.

#### 4.1.2 Database for Global Authentication

We use MS Passport [8] as our authentication database. Several factors motivated our choice of Passport as the authentication service. 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 *https*. 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.

### 4.1.3 The PANS Authorizer

The job of the *PANS Authorizer* is to authorize client's access to the network upon successful completion of user authentication. Additionally, it handles the task of determining service policies, generating per-user keys, and communicating keys to the clients and to the service gateways (*PANS Verifiers*, to be discussed next). 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.

Upon authentication, the authorizer looks up its policy table to determine the user's service level, generates a (*key_id, key, token*) triple, and then communicates this to the *PANS Client* module residing on the user's device and to the service gateways. The gateways are given the key-triple, which is stored into an array of valid (key, token) pairs indexed by the key_id assigned to the clients. Once the user has been authenticated, all her communication is directed to the assigned service gateway. Individual packets are key-tagged by the client and verified by the service gateways to ensure that only authorized traffic is allowed to gain access to the Internet

Each (key, token) pair is valid for a finite amount of time. A user session is separate from key expiration times. 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.

### 4.1.4 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 task of the authorizer and the verifier are separated out in order to achieve a clean separation in the time scales of their operation. The tasks performed by the authorizer are as frequent as the number of new authentication operations, either due to new users in the network or due to an old user having timed out. On the other hand, the PANS Verifier actively processes each packet that is sent out of the mobile host and runs on a much smaller time scale as compared to the authorizer. The task of the PANS Verifier includes checking if each packet from a client (identified by a unique key_id) contains the right (*key, token*) combination that the PANS Verifier has in its table entries. 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. The separation between authorizer and the verifier was additionally motivated by the need to support a greater number of PANS Verifiers as we introduced support for roaming. This would mean replication of the PANS Verifier; one for each subnet of wireless access points.

### 4.1.5 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.

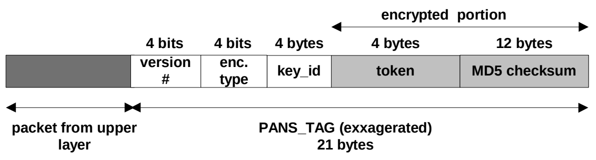


**Figure 2**: The PANS_TAG showing the different fields. The version number, encryption type and `key_id` form the unencrypted portion, while the token and MD5 checksum are encrypted using the encryption algorithm specified under the encryption type.

Packets are tagged 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.

### 4.1.6 The PANS Announcer

To facilitate auto-configuration of the PANS Client module, PANS implements a set of Announcer and Client Configuration modules, which resides on the PANS Authorizer and on the PANS client respectively. The Announcer periodically broadcasts a beacon, which contains an unique PANS network_id, a subnet_mask, and a pair of authorizer_ip and verifier_ip addresses to identify the PANS Authorizer and PANS Verifier that are to be used in the local network or wireless access point (AP). The Client Configuration module uses the beacon information to determine whether the client has migrated to a different network and to set the client's default gateway in order to direct traffic to the correct PANS server.

Having described the key system components of the CHOICE network we are now ready to discuss our solutions to the mobility problem described in section 5.

.

## 5 Dealing with the Mobility Problem

In this section we describe the design criterion, the architecture, algorithms and the implementation details behind the mobility support module that we have built for the CHOICE network. We then describe how this module helps us combat the mobility problem by running through different scenarios

## 5.1 Design Criteria

The goals that influenced the design of the mobility support 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 lightweight and efficient in terms of bandwidth, memory, processing, and power.

*(II) 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.

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

*(IV) Versatility:* We wish to examine whether employing a particular scheme would benefit or help ease the design of other components of the PANS system.

In the wireless network, a beaconing scheme is generally more efficient than a polling-response scheme due to the following reasons:

- Beaconing is unidirectional so it cuts transmission overhead by half for wireless hosts when compared to a bi-directional polling-response scheme.
- Beaconing consumes only one unit of airtime per broadcast period, compared to the 2n units of airtime used by all polling and response messages created by n different clients in a wireless network. Hence, beaconing reduces both the total amount of airtime overhead and the level of traffic contention.
- Polling can introduce additional waste in private networks as the client periodically sends broadcast packets to probe for the PANS service. As an alternative to periodic polling, a client can send a limited number of broadcast queries only when necessary; that is, when there is a good hint that the client has migrated to another network (e.g. the hardware detects a link state change or when the host detects excessive amount of packet lost, see black-hole detection [10]. Unfortunately, such advice given by other network layers are often implementation dependent and consequently, unreliable. Please refer to section 8.4 for further discussions.

## 5.2 Architecture and Implementation

Due to the considerations listed above, we have designed and built an auto-configuration system that allows the client host to discover the PANS service and receive configuration parameters via broadcast advertisements or beacons. The client then uses the beacon information to configure itself and launches the authentication process to gain network access. When the client migrates out of the public network service, it no longer receives any beacon. In this case, the client waits for a timeout period and resumes normal networking operation by disabling the special mode at the local PANS driver.

Our scheme is very similar to the broadcast advertising schemes found in Mobile IPv4 and Mobile IPv6 [15] 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.

### 5.2.1 Flow Control

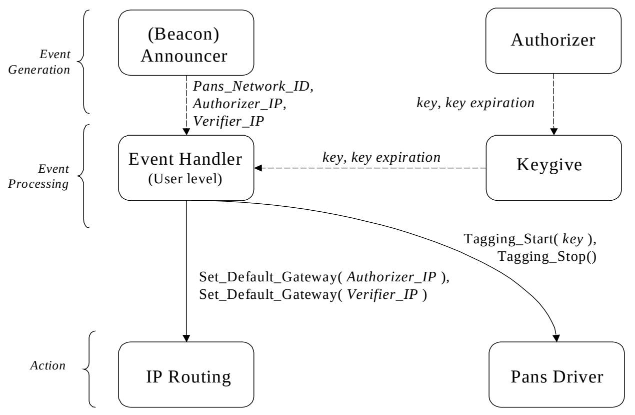

Event Generation
Event Processing
Action
(Beacon) Announcer
Authorizer
Pans_Network_ID, Authorizer_IP, Verifier_IP
key, key expiration
Event Handler (User level)
key, key expiration
Keygive
Tagging_Start( key ), Tagging_Stop()
Set_Default_Gateway( Authorizer_IP ), Set_Default_Gateway( Verifier_IP )
IP Routing
Pans Driver

**Figure 3:** Architecture for auto-configuration of PANS

Figure 3 illustrates the architecture for supporting auto-configuration in PANS. The diagram divides the software modules into three separate classes: *event generation*, *event processing*, and *action*. The Beacon Announcer module in the *event generation* class runs on the PANS authorizer server in the public network. As mentioned earlier, the Announcer periodically broadcasts a beacon, which contains a unique PANS network_id, a subnet_mask, and the current authorizer_ip and verifier_ip addresses to identify the pair of PANS Authorzier and PANS Verifier that are to be used in the local network or wireless access point (AP). The Event Handler, a component of the Client Configuration module installed on the client host, uses the network_id and the subnet_mask to distinguish whether the client has a) roamed to a different subnet within the same network or b) migrated to a different public network altogether. The Event Handler uses the authorizer and verifier IP addresses to set the client's default gateway in order to direct traffic to the correct server, according to the current state of the PANS protocol.

The PANS Authorizer, as mentioned in Section 2, serves as a proxy for a global authentication server such as MS Passport. Whenever the user completes an authentication process, the Authorize delivers the key-triple and key-triple expiration values via a secure *https* connection, which is established by the client's web browser when the user logs on to the Authorizer web server. To do this, the ASP script on the web server delivers the key values via a MIME-typed data stream, which triggers the web browser to launch the registered Keygive program. The web browser then pipes the key values to the Keygive module, which in turn hands them over to the Event Handler.

The *event processing* modules are the Keygive module and the Event Handler. As illustrated above, the Keygive module acts as a web browser relay to pass the key values to the Event Handler. The Event Handler is a user-level daemon that performs the auto-configuration of the network parameters and key management on the client host. It listens to two well-known ports: one is for detecting beacons coming from the Announcer, and the other is for receiving the key-triple and key-triple expiration values from the Keygive module. The Event Handler stores the key values into a table indexed by the network_id. The Event Handler implements an "earliest-expiry-time" (EET) replacement policy[1] and invalidates a key-triple entry whenever it expires. Then by matching a valid row entry with the currently advertised network_id, the Event Handler can configure the local PANS driver with the appropriate key-triple value via a local ioctl call.

The *action* modules consists of the local PANS driver and the client's route table. The Event Handler issues ioctl calls to pass the key-triple, and to enable / disable the PANS driver and system calls to set the default gateway in order to direct the client host traffic. A detailed explanation of the interactions between these components is described in the upcoming section.

Notice that the verifier_ip value is deliberately transmitted inside the broadcast beacon instead of being transmitted alongside with the key-triple and key-triple expiration values. This is done to allow those clients who have migrated to a different subnet but still hold a valid key-triple to directly access the local verifier[2] without repeating a full authentication process. Furthermore, such a design supports a useful system-wide fail-over feature. For example, both the authorizer_ip and verifier_ip can be changed dynamically to immediately migrate all the clients to use another set of default gateways.

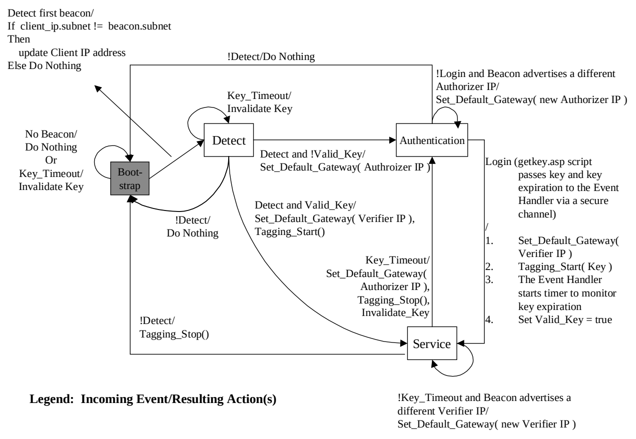
Thus by including all the network parameters within the beacon, we have decoupled key management and mobility management so that the network access protocol (PANS) and the auto-configuration mechanism can work independently of each other. In our system, the client is not required to re-negotiate a key that has not been expired, regardless how many times a client has moved between different networks. Likewise, the authorizer is free to refresh a client's key (via out-of-band mechanisms) during an authenticated session without affecting the client host's network configuration. In Section 6, we will explain how this decoupling of key and mobility management also helps us design a scalable key management scheme for the public network infrastructure.

[1] The replacement policy is arbitrary. If there is a monetary cost associated with obtaining keys, a better metric might be a policy that minimizes the key replacement cost.

[2] During the authentication process, the client key may be distributed globally among the verifiers located in different subnets. A scalable key distribution mechanism is discussed in Section 7.

### 5.2.2 The Event Handler Algorithm

Figure 3 shows the finite state machine implemented by our Event Handler. A typical PANS session begins in the Bootstrap state, where the Event Handler determines if the PANS service is available. If no beacon reaches the client, the Event Handler remains in the Bootstrap state, no action is performed and the client communicates with the network normally with the PANS driver disabled.



Detect first beacon/
If client_ip.subnet != beacon.subnet
Then
update Client IP address
Else Do Nothing
!Detect/Do Nothing
!Login and Beacon advertises a different Authorizer IP/
Set_Default_Gateway( new Authorizer IP )
Key_Timeout/
Invalidate Key
No Beacon/
Do Nothing
Or
Key_Timeout/
Invalidate Key
Boot-strap
Detect
Authentication
Detect and !Valid_Key/
Set_Default_Gateway( Authroizer IP )
Login (getkey.asp script passes key and key expiration to the Event Handler via a secure channel)
/
1. Set_Default_Gateway( Verifier IP )
2. Tagging_Start( Key )
3. The Event Handler starts timer to monitor key expiration
4. Set Valid_Key = true
!Detect/
Do Nothing
Detect and Valid_Key/
Set_Default_Gateway( Verifier IP ),
Tagging_Start()
Key_Timeout/
Set_Default_Gateway(
Authorizer IP ),
Tagging_Stop(),
Invalidate_Key
!Detect/
Tagging_Stop()
Service
Legend: Incoming Event/Resulting Action(s)
!Key_Timeout and Beacon advertises a different Verifier IP/
Set_Default_Gateway( new Verifier IP )

**Figure 4:** State Transition Diagram for the Event Handler

When the Event Handler detects the first beacon, it pops up a greeting message to notify the user about the discovery of the PANS service. If the user wishes to join the service, the Event Handler verifies that client has a valid IP address to operate in the subnet currently advertised in the beacon and updates the address if necessary. Our system currently relies on DHCP to obtain a dynamic address assignment. To ensure timely address assignment, the Event Handler will force a DHCP request and loop in the Detect State until the client host receives a valid address. This is done to handle roaming problems caused by inconsistent media sensing implementations. See section 8 for a discussion.

Next, the Event Handler checks the key table if the client currently posses a valid key for the current network. If so, it bypasses the login and authentication procedures and enters the Service state so that the client can seamlessly resume the previous PANS session. If the client does not have a valid key, the Event Handler enters the Authentication state where it sets the client's default gateway to the advertised Authorizer IP address. At the same time, the user launches her default web browser and directs it to the default site <u>*http://choice*</u>[3], which is hosted on the Authorizer web server. The user then performs the web-based authentications process. The Event Handler waits in the Authentication state until the authentication succeeds or until the client migrates out of the current network.

After successful authentication, the Event Handler obtains a set of key values and enters the Service state. The Event Handler finds an empty row entry in the key table (with EET replacement policy) and inserts the network_id, the key-triple, and

[3] Recall that the Authorizer grants clients restricted access only to the public authentication server, the authentication proxy, and basic level services such as DNS, WINS and any other free services offered by the service provider.

key_expiration_time into the table. At the same time, it issues an ioctl to pass the key to the PANS driver, trigger the driver to start tagging packets, and sets the default gateway to the advertised verifier_ip. At this point, the client gains access to the rest of the network by sending legitimate (tagged) packets through the verifier. The Event Handler remains in the Service state until either the key expires or when no beacons are detected for a timeout period. The former case would return the Event Handler to the Authentication state to allow the client to re-negotiate a new set of key and key expiration values with the Authorizer. The latter case might indicate that the client has migrated outside the PANS service area. Hence, the Event Handler returns to the Bootstrap state and stops the PANS driver from tagging any packets. Note that the client maintains the key until it expires so that it could resume the ongoing session when the client returns to the PANS service area (i.e. when it starts detecting beacons again).

Finally, we wish to explain the self-loop transitions in both the Authentication and Service state. These transitions occur whenever the Event Handler detects a change in the beacon. A change in the network_id signifies that the client has migrated to a different subnet or a different public network. In this case, the Event Handler jumps back to the Detect state (transitions omitted from the diagram for clarity). A change in either of the advertised authorizer_ip or verifier_ip will cause the Event Handler to reset the default gateway. This feature will facilitate the fail-over and load-balancing mechanisms discussed in section 6.

## 5.3 System Operation in the face of Mobility

This section describes how mobility is handled when a client migrates between a foreign network and the PANS network, and between subnets within the same PANS network.

### 5.3.1 Inter-Network Mobility – Moving Between a Private and Public Network

A client might enter a PANS network without an IP address (e.g. the client is turned off before entering the PANS network) or with a stale IP address (e.g. the client obtains an Auto IP, or the client hibernates before disconnecting with the external network). In either case, the client entering the network would eventually receive a beacon and detects the existence of the PANS service. At this point, the client Event Handler checks if the subnet of the client host address matches the one advertised by the beacon. If the address is found stale (i.e. the subnets do not match), it triggers DHCP to obtain a dynamically assigned IP address. Afterwards the Event Handler sets the default gateway to the advertised Authorizer IP address. Now, the client proceeds with authentication and normal operation within the PANS service.

When the client moves *out* of the network, no beacon reaches the client. When the Event Handler times out, it stops the PANS driver from tagging any packets. Thus, the client can attach to any foreign network, requests an IP address update (via DHCP), and resume normal operation. The client maintains the key until it expires. When the client returns to the PANS service, it can resume the most recent PANS session using the key it saved.

### 5.3.2 Intra-network Mobility – Moving between Subnets within the Public Network

In this scenario, the client has either entered the PANS network but has moved into another subnet without completing the authentication process or already initiated a PANS session. In either case, the Event Handler detects that it is receiving a beacon containing different subnet information. Consequently, it would first update the client's IP address, and then assign the default gateway to the advertised Authorizer or Verifier address, depending whether the client already has a valid key for the ongoing session.

## 6 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.

## 6.1 Fail-over Mechanism to Provide High Availability Service.

Because the authorizer and verifier contain the set of active key values in the network, the public network service must provide a fail-over mechanism to handle the case when either of the authorizer or verifier machine fails. To prevent loss of information, a service provider may install multiple redundant verifiers that replicate the table of keys currently active in the network. Thus, whenever the authorizer detects a verifier failure, it can announce a different beacon containing the address of the redundant verifier. (A similar mechanism can be used to fail-over the authorizer.) Upon receiving the new beacon, every client in the network will immediately change its default gateway to the backup verifier. Because the backup verifier contains the set of all active keys in the network, the transition should occur smoothly, with minimal disruptions, if any, to all ongoing network transaction.

## 6.2 Load Balancing

In addition to providing a highly available service, multiple verifiers can be installed within a single network (or sub-network) to load-balance the client traffic. Here, a query-response mechanism would work very nicely because each client would query a configuration server (possibly the authorizer) to assign a verifier for all its network transactions. Then to load-balance traffic, all the configuration server needs to do is to respond to each client with a different verifier assignment in rotation.

Unfortunately, we will have to design a formal protocol specification to support such a query-response mechanism, which would complicate our system. Moreover, such a scheme is not compatible with our fail-over mechanism described above because when a particular load-balanced verifier fails, only a subset of clients are affected. Hence, any fail-over mechanism will need to figure out a) which clients are affected and b) how to notify the affected clients and c) how to migrate the affected clients to the appropriate backup server. Such a fail-over mechanism can become very complex.

Our proposed approach to load balancing is particularly well-suited to wireless public networks[4]. The public network can load-balance by assigning each authorizer, verifier and the associated backup server to handle only the traffic coming from a subset of wireless access points (AP). Hence, clients in different subsets of APs would access a different set of authorizer and verifier servers. To support this mechanism, each authorizer must *multicast* beacons to its own subset of APs. The APs in turn, must broadcast the beacon to all of its associated clients. In this scheme, the fail-over mechanism would work the same way as described above.

Unfortunately, there is currently no mechanism that allows the system to *multicast* broadcast packets to a subset of specific access points in a wireless network. Although one of our original design philosophy is to avoid changing any existing hardware or software protocols, we wish to advocate for implementing such a feature into future wireless networks. We believe this feature would not require a significant change to the existing wireless infrastructure. We also believe that other beacon-based wireless applications would greatly benefit from this feature.

## 6.3 Scaling and Subnetting

Another simple way to scale a large public network is to divide it into subnets. This is similar to load-balancing, where each subnet would have its own set of authorizer, verifier and backup servers to share the load of the network. However, the key difference is that the network has explicitly been divided into separate address spaces. Consequently, the client must now change its IP address, which would disrupt any ongoing network applications running at the client host[5]. On the other hand, subnetting may be required for a number of administrative reasons. Hence, the decision is left to the service providers to choose the appropriate combinations of load-balancing and subnetting to meet their needs.

[4] At this point, we are not able to offer a simple load-balancing solution for wired networks. We will leave this as part of our future work.

[5] The user should use protocols such as Mobile IP or TCP Migrate to migrate its applications when the client changes network attachment points (i.e. IP addresses).

In any case, a public network must find a scalable and efficient method to manage client keys when the clients are mobile. When a mobile client roams within a load-balanced network or migrates across subnets, the host must change its default gateway configuration to direct is traffic to a new verifier and use the same key to gain access in the new network location. In section 5, 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 according to the client's current location.

Here, the auto-configuration system can help. The Event Handler can keep a history of the subnet that the client has previously visited. Then when the client roams to another subnet, the Event Handler 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 are the key_id index and the address of authorizer that had issued the key. The authorizer in the new subnet would forward the request to the indicated authorizer, which authenticates the request by checking the encrypted token. If the verification succeeds, the authorizer in the old network will send the requested key to the authorizer (via secure channels) in the new subnet. After the new authorizer receives the key, it distributes the key among the verifiers in the new subnet and acknowledges the client's Event Handler. Thus, the client migrates to a new subnet seamlessly by using this scalable, on-demand approach to key distribution.

Certainly, a more sophisticated form of migration negotiation policy (such as restricting certain clients from accessing certain subnets) can be easily added to the architecture described above.

## 6.4 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.

### 6.4.1 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 Event Handler can be modified to interpret these metrics and notify the user appropriately. 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, it would be desirable to find a way to extract load information from the individual APs programmatically.

#### 6.4.2 Coarse level Location Information

In the wireless network, one could imagine that every access point broadcasts a beacon that includes an access point identifier. Then during the authentication process, the authorizer may download a graphical map showing the user's current location with respect to her access point association. Although this is a very coarse-grain approach for identifying the user's location when compared to the proposed alternatives [28][29][30], it is nevertheless a useful feature (especially in large settings such as the airport) 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 Event Handler can map the identifier to a table of messages. Thus, depending on the user's preference, the Event Handler 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 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 to be abused for implementing heavy-duty service discovery applications mentioned in [26][27]. 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 shows how to do this by means of mapping compact codes contained in the beacon with a table containing the full information required for the application.

Supporting location information services in our auto-configuration system may require specifying an extensible beacon packet format to include option fields much like those found in DHCP. Although we have not currently implemented such a feature into our own system, we believe doing so should be relatively straightforward.

## 7 Discussions

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

### 7.1 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 typing our system to any specific protocols. Hence, any protocol, including Mobile IP, will continue to operate seamlessly on our system.

### 7.2 Low-Level Configuration

There is one situation that prevents our auto-configuration system from migrating a client between the public and private networks. The problem is caused by special configurations in the wireless network interface (WNIC). As mentioned in the introduction sections, some private networks uses the wireless equivalency protocol (WEP) to secure the wireless link. Because WEP did not provision for mobility support, clients who wish to communicate with a WEP network must manually configure and enable the WEP settings in the WNIC driver. We attempted to modify our auto-configuration system to address this issue. However, we discovered that none of the IP-level broadcast packets would reach the client unless WEP is probably configured. Hence, we concluded that such low-level type of configuration must be done by the protocols (i.e. WEP) at the same level.

Nevertheless, we do not believe this to be a huge concern as implementing mobility extensions to WEP should not be a difficult task. With the appropriate extensions and API hooks for supporting mobility in WEP, our system should migrate clients seamlessly between all types of public and private networks.

### 7.3 High-Level Configuration

Although this is more of a minor annoyance than a serious problem, we note that some applications will need to be reconfigured 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 must be made aware of the host's mobility. In order to do this, operating systems should facilitate API callbacks to notify applications appropriately.

We should mention that another solution to this problem is to employ Mobile IP. However, using such techniques may imply certain limitations [31]. 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 corporate proxy and its access policies. In contrast, the user may gain full access to the Internet via other end-to-end mobility mechanisms that allows applications to open direct connections and assume the access policies defined by the public network.

### 7.4 Timely Mobility Detection: 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 [15].

Using this mobility detection scheme, there is a tradeoff between response time and beaconing overhead. In order to reduce the detection latency, a network may increase the beaconing frequency. But increasing the beacon frequency increases the overhead of the system.

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 nor the extended use of polling. That is, the polling stops as soon as the DHCP server responds to the client's message. There is no need to rely on polling nor beaconing as a "keep-alive" signal because the next link-state change would notify 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 well-defined media sensing capabilities across different network interface technologies and their implementations have reinforced our design goal to be hardware agnostic. Hence, we have used beacons to facilitate mobility detection in our auto-configuration mechanism. We should note that the cost for timely response due to the increase in beaconing frequency 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 a 11Mbps wireless network.

### 7.5 Security

A good question to ask when examining the design of any auto-configuration system is how well does the system handle 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[6].
- The client can use the full packet encryption feature provided in PANS to increase security.

The auto-configuration system uses the beacon to trigger client host configuration. Hence, the beacon becomes the entry point for all 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.

#### 7.5.1 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, we should make it difficult and 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 the attack by increasing the attacker's exposure to the system, thus preventing the 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. In this case, a malicious attacker can fool a receiver by replaying short-range beacons towards it. However, the attack must devise such a device and possibly leave more traces of evidence about the attack.

#### 7.5.2 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 encrypted the challenge

[6] The reader may recall the key-deliver mechanism by which the Authorizer passes the key values via a MIME-typed data stream. Upon receiving this data, the browser than launches the Keygive program, which passes the key to the Event Handler. This appears to be a weak security link in our system, where the client may visit a malicious web server and trigger the Keygive program by delivering a harmful MIME-typed data stream. Although not in our current implementation, we could easily modify the Keygive program to perform authentication on the MIME-typed data stream via certificates or other out-of-band mechanisms.

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. Also, the Event Handler can detect some forms of hijacking attacks by delaying the migration process and compare the beacons for a small period of time. If it consistently hears two different beacons during that period, it can alert the user of a possible hijacking attack. Randomizing the beacon intervals will also help strengthen the detection of the hijack attack. The attacker must either jam the entire wireless channel or intercept the packet in the physical network in order to fool the detection system.

## 8 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 [16],[17], and [20]. 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 [17]. 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 [18]. 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) [21], IPsec encapsulating security payload (ESP) [22] and IKE [23] 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 realms of supporting mobility, there is Mobile-IP (v4 and v6) [15], 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 [27] and MIT INS [26] 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 an over-kill for our application.

In the realms 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.

To summarize, 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 in order 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.

## 9 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 unaware 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. The fact 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. Although in some cases trivial the existence of standards in device programming and access point programming can help us achieve our ultimate goal of providing seamless mobility.

## References

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

[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] P G. Viscarola and W. A. Mason, Windows NT Device Driver Development, *OSR Open System Resources*, 1999

[12] CCITT.Recommendation X.509: The Directory-Authentication Framework, Geneva, 1989

[13] VeriSign Inc., Internet Trust Service http://www.verisign.com/

[14] R. Stine, "FYI on a Network Management Tool CatalogTools for Monitoring and Debugging TCP/IP Internets and Interconnected Devices," *IETF RFC 1147*, April 1990, http://www.ietf.org/rfc/rfc1147.txt

[15] 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

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

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

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

[19] C. Rigney, A. C. Rubens, W. A. Simpson, S. Willens, "Remote Authentication Dial-In user Service (RADIUS)," IETF RFC 1238, http://www.ietf.org/rfc/rfc128.txt

[20] 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

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

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

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

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

[25] The Wireless Application Protocol (WAP) White Paper, http://www.wapforum.org/what/whitepapers.htm

[26] 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.

[27] 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

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

[29] R. Want, A. Hopper, V. Falcao and J. Gibbons, "The Active Badge Location System." *ACM Transactions on Information Systems* 10, 1 (January 1992), 91-102

[30] N. Priyanth,, A. Chakraborty, H. Balakrishnan, "The Cricket Location-Support System," In Proceedings of the ACM/IEEE MOBICOM 2000 (Boston, MA, Aug. 2000).

[31] S. Cheshire, M. Baker, "Internet Mobility 4x4." In *Proceedings of SIGCOMM 1996*, August 1996