Policy-Enabled Handoffs Across Heterogeneous Wireless Networks

Helen J. Wang
Computer Science Division
University of California, Berkeley

December 17, 1998

1 Abstract

"Access is the killer app" [3] is the vision of the Daedalus project at U.C. Berkeley. Being able to be connected *seamlessly* anytime anywhere to the *best* network still remains an unfulfilled goal. Often, even determining the "best" network is a challenging task because of the widespread deployment of overlapping wireless networks. In this report, we describe a *policy-enabled handoff system* that allows users to express policies on what is the "best" wireless system at any moment, and make tradeoffs among network characteristics and dynamics such as cost, performance and power consumption. We designed a performance reporting scheme estimating current network conditions, which serves as input to the policy specification. A primary goal of this work is to make it possible to balance the bandwidth load across networks with comparable performance. We identified the problem of handoff instability that may be caused by handoff synchronization, i.e., the scenario of many mobile hosts making the same handoff decision at essentially the same time. We use randomization to break such synchronizations. Given the current "best" network, our system also determines whether the handoff is worthwhile based on the handoff overhead and potential network usage duration.

2 Introduction

Katz, et al. [8], has viewed today's collection of infrared, radio wireless LAN, cellular and satellite networks as an overlaid structure of room-size, buildingsize, and wide area data networks, which are termed *Wireless Overlay Networks*. The "goodness" of these networks fall into a total ordering; namely, the lower the level of overlay, the smaller the coverage area it has, and the higher bandwidth per mobile host it offers. Because of the widespread deployment of wireless networks and new emerging wireless technologies, we see the emergence of a topological order of wireless networks, where a number of networks offer similar coverage and bandwidth on the same overlay level. It is not obvious which network is better without considering dynamic conditions, such as current traffic load, cost and power consumption of the network usage.

"Vertical handoff" [13] describes a mobile host roaming across wireless overlay networks and its implementation with three overlays: Infrared in-room LAN, in-building WaveLAN, and Metricom Ricochet wide area wireless networks[8]. It used Mobile IP [6] focusing on minimizing handoff latency. "Seamlessness" is achieved in the sense that handoffs are unnoticeable. The vertical handoff decision is simple, and is embedded in the system. The mobile host always switches to the "lowest" (smallest coverage area) reachable overlay. Unfortunately, this policy ignores system dynamics. In addition, one fixed policy complicates the adoption of new networks, especially those with similar coverage area and bandwidth.

In this report, we present a *policy-enabled handoff system*, which separates the decision making (i.e., what is the "best" network and when to handoff) from the handoff mechanism. This flexibility is required in light of plethora of emerging wireless WANs, such as GPRS [1], WCDMA [16], Infostation [4], and satellite networks. Policies on what the "best" reachable network is, and when to handoff to it, can be complex to specify. A single, hard coded policy is suboptimal.

We will describe the operating environment of our policy-enabled handoff system, which has a Mobile IP infrastructure. Then, we present policy enabling mechanisms including a dynamic network condition estimation scheme. We experimented with a cost function-based policy model. We also identified the issue of system stability, and designed hysteresis into the handoff mechanism.

Our testbed includes networks based on IBM Infrared LAN, Lucent WaveLAN, Metricom Ricochet, and GSM Cellular Modem. Nonetheless, it is network independent. Any network in any numbers can be used in the system. We choose these networks because they illustrate an interesting "goodness" ordering. IBM Infrared LAN is on the lowest overlay with smallest coverage area and highest bandwidth per coverage area; WaveLAN is on the second overlay network with wider coverage area and lower bandwidth per coverage area than IR; and both Metricom Ricochet and GSM Cellular are on the third overlay with similar bandwidth but lower than WaveLAN's bandwidth, and both have wide area coverage.

The rest of the report is organized as follows: We first look at the related work (Section 3). Then we motivate the case for *Policy-Enabled Handoff System* and address its design principles (Section 4). We describe our operating environment in Section 5. Then, policy specification is discussed and presented in Section 6. Handoff synchronization occurs when several mobile hosts make handoff decisions synchronously causing dramatic load increase or decrease on the involved networks. We propose the solution for this problem in Section 7. Implementation techniques and software architecture are described in Section 8. We evaluate our system in Section 9, discuss the future work in Section 10, and finally conclude in Section 11.

3 Related Work

Mobile IP [6] has been proposed to solve the mobility problem. The essential mobility problem lies in the dual roles of an IP address as both the identity and the physical location of a host. Mobile IP separates these by dedicating different IP addresses for different roles. The *home address* of a mobile host (MH) is its identifying IP address. Its *care-of* address refers to the temporary IP address at a foreign network it is visiting, and therefore indicates the current physical location. To get packets destined for a MH to its care-of address, a *home agent* introduces a level of indirection, keeping track of the current care-of address. MHs send location update messages to their home agents whenever they move to a location with a new care-of address. Home agents then route the packets to a MH's care-of address through IP in IP encapsulation (i.e.,*tunneling*). Base stations are responsible for delivering packets to MHs over the last wireless hop.

Seshan et al. [10] described a scheme to achieve low latency handoffs across cells using multicast as another level of indirection. Its usage prevents the adverse effect of TCP congestion control due to packet loss during handoffs. More than one base stations receives packets for the mobile host with one base station forwarding and the rest buffering. Buffering before actively forwarding at the base stations reduces handoff latency. Also, the use of the multicast address eliminates location updates from the mobile host to the home agent. The vertical handoff system and the system described here adopted multicast care-of address for the same reason. In addition, we allow base stations from different networks to listen on the same multicast care-of address.

Stemm and Katz [13] introduced the term *vertical handoff* for handoffs across two cells from different networks, while *horizontal handoff* refers to the handoff between cells within the same network. The vertical handoff system is the precursor of our work. It explored many techniques such as fast beaconing and header doublecasting to reduce the handoff latency. Nonetheless, the system used a single handoff policy, and did not address the issue of more general policies.

MosquitoNet[17],[2] also addressed the Mobile IP policy issues. While our policies focus on choosing the "best" network, their policies focus on choosing the most desireable packet delivery path based on the characteristics of traffic flows. Their work also enabled simultaneous use of multiple network interfaces, which our infrastructure does not support.

In the next section, we motivate the need for policy-enabled handoffs.

4 The Case and Principles for Policy-Enabled Handoffs

Many wide area wireless network technologies are emerging. Wideband CDMA [16] is designed to meet the future requirements of the third generation wireless communication services with data rates up to 2Mbps. Both packet and circuit switched services can be freely mixed, with variable bandwidth, and delivered simultaneously to the same user with specific quality levels. GPRS (Generalized Packet Radio Service) is a soon-to-be available packet data service within GSM allowing bit rates from 9 to more than 150kbps [1]. The user will be charged for the amount of data that is transferred and not for the connection time, as in the circuit switched case. Table 1 enumerates the diverse charge models for different wireless networks.

Infostations [4], in analogy to gas stations, provide high network bandwidth for users to fill up their information tank. Satellite networks also promise "global coverage" and "Go-anywhere reliability" (i.e., Spaceway [12]

Wireless Network	Charge Model
Cellular Modems (Circuit Switched Data)	Function of connection time
Highspeed Circuit Switched Data	Function of connect time and data rate
GPRS, WCDMA (Packet Switched)	Function of data transmitted
Metricom Ricochet	Flat monthly rate
Wireless LANs (IR, WaveLAN)	Free of charge beyond the fixed cost of infrastructure

Table 1: Charge Models for Wireless Networks

and Teledesic [15]). Existing wide area wireless networks include Metricom Ricochet wireless modem, CDPD, and cellular modems. All these network technologies differ in bandwidth, latency, power consumption and potentially their charge model. The issue is how to integrate these *seamlessly*.

In addition, many have envisioned the future wireless networks as having a common core network integrating different network access technologies. The Iceberg project [7] at U.C. Berkeley is developing a testbed with an Internet-based core network backbone and different network access technologies such as Infrared, WaveLAN, GSM, CDMA, etc. Policy-enabled handoffs are essential.

Also, dynamic factors must be considered in handoff decisions for effective network usage. For example, information on current network conditions can help load balancing across networks; current user conditions, such as a mobile host's moving speed can eliminate certain networks from consideration (i.e., those networks that do not support mobility). Available hints, like user activity pattern and network coverage maps, can also contribute to handoff decisions.

We need a *policy-enabled handoff system*. We maintain the same goal of mobility: *seamlessness*. To achieve this, first, handoff latency must be low enough to not disrupt the running applications. This issue has already been addressed by [10] and [13]. Second, the automation of switching from one network to another is based on the principle of *user involvement with minimal user interaction*. User involvement is required for the policy specification, while minimal user interaction implies automation. It is essential that policy specification be simple and intuitive, otherwise users will manually configure the network, thus violating our goal of *seamlessness*. In Section 6, we will present our policy specification model that involves the user but minimizes user interactions.

5 Operating Environment

The operating environment of *policy-enabled handoff* is a Mobile IP-like infrastructure, as shown in Figure 1. When a correspondent host sends packets

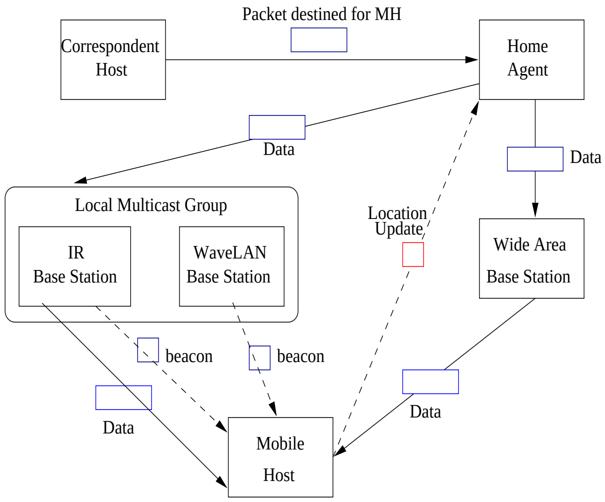

Figure 1: The system setup

to a mobile host, they go through its home agent. The home agent routes the packets either to the multicast care-of address (the local multicast group in Figure 1) or to the unicast care-of address of the mobile host. When the mobile host is in a wireless LAN, the multicast care-of address is used for the same reason as in [10]. We are unable to make Wireless WAN base stations participate in the multicast group because they are out of our control. In this case, we require the mobile host to send location update messages to the home agent as in conventional Mobile IP. Consequently, the home agent needs to keep track of whether the mobile host is in a wireless LAN, its multicast care-of address and its unicast care-of address. The use of multicast care-of address does pose security concerns; namely, any node within the TTL of the home agent can subscribe to the group, allowing it to eavesdrop on traffic that it would not otherwise be able to receive.

Handoff decisions and operations are all done at the mobile host. Periodically, the mobile host collects current dynamic conditions, and consults with a policy module on which is the "best" reachable network. If it is not the one in use, and it has been consistently the "best" for a period of time, the mobile host hands off to it. The handoff operation involves routing table manipulation, and sending location updates.

This operating environment is similar to Stemm's [10] vertical handoff system except that we use conventional Mobile IP when the mobile host is in WAN.

The next section illustrates how policies are specified.

6 Policy Specification

In this section, we describe policy considerations in the face of handoff decisions. Then we introduce our policy model, and some policy examples we experimented.

6.1 Policy Parameters

Policies are specified in terms of the following parameters: cost, network conditions, power consumption, connection duration time, connection setup time, and various hints. We explain each in turn.

There are many charge models (see Table 1). Wireless LANs are typically free of charge. Metricom Ricochet wireless modem has a flat monthly rate. Many connection-oriented networks impose static or variable charge per time. Packet switched networks tend to charge per data amount. Networks that provide Quality of Services may differentiate and charge differently for their services. The INDEX project at U.C. Berkeley experiments with such a network seeking for a proper billing scheme[5]. Users are the ones who pay, and must be allowed to specify when to pay how much for their desired services.

Network condition is a dynamic parameter in choosing the "best" network. Current performance of reachable networks is a factor in determining the cost performance tradeoffs. That includes available bandwidth at each reachable network, network latency, and reliability of the reachable networks. The latter can be characterized by the number of retransmissions. When a high bandwidth network, say Infrared LAN, is heavily loaded or congested, it may be wiser for the mobile host to switch to a lower bandwidth one, say Metricom Ricochet, with little traffic.

Power consumption is another dynamic factor. Wireless transmission can be performed through PCMCIA interface cards (e.g., IR, WaveLAN, cellular modem card) or other devices. While PCMCIA cards acquire power from the mobile host's battery, other devices have an external battery supply (e.g., Ricochet modem). Furthermore, although many wireless networks are accessed through PCMCIA cards, they have different power consumptions because network usage varies among them. When the mobile host's battery is low, the mobile host may choose the reachable network with the least power consumption. For example, Metricom Ricochet may be chosen over WaveLAN if the Ricochet's external battery has enough power. Similarly a low external battery of a network device can eliminate that network from handoff consideration.

For connection oriented networks, connection setup time is long and may impose extra charges if one is just passing by this network quickly. The connection duration will be short, thus switching[1] to this network may not be worthwhile.

Many hints like user activity history, current speed of the user, network coverage maps can also drive the handoff decisions. For example, a driver would want to choose the GSM cellular network rather than Metricom Ricochet because the latter cannot accomodate vehicle speeds. If the user speed and moving direction is known, with a network coverage map, it is predictable when the user will leave a network and enter another one. This information can help hide handoff latency by initiating handoff before disconnection from the former network.

[1]We treat the connection setup time as part of the handoff latency.

6.2 Policy Enabling Mechanisms

In this section, we illustrate the mechanisms needed to enable policy specifications.

As in many microkernel design principles, the flexibility and efficiency of a system lies in exposing the system internals while enabling customization through specified policies. This requires defining a system API for accessing system status while performing system operations. System status includes the current network and base station in use, and dynamic information collected by the system such as current network loads on all reachable networks[2] (we will describe our scheme of estimating the network performance in detail in the 6.2.1), observed throughput, how long the user has used each network, how many bytes the user has transferred on each network, the current battery status on the mobile device, and how much money the user has spent on each network. System operations include the handoff operation that manipulates the routing table to switch to a new network.

Some additional information is needed for policy specification. It includes network characteristics, such as whether a network is connection oriented (if yes, what the connection setup time is), the typical bandwidth or latency it offers, its power consumption, and its charge model and rate. Table 2 shows the network parameters and their values for the four networks in our prototype. This information is stored in our network database object described in Section 8.2.

User policies, parameterized by this information, determine the "best" network. This is determined periodically. If a different network has been consistently better than the current network, then the system triggers handoff. We explain the meaning of "consistently" in more detail in Section 6.3.2.

6.2.1 Estimating Network Conditions

Estimating network conditions can enable policies. We have devised a scheme to estimate network load for those networks under our control[3]. We implemented a performance agent that collects the information on current bandwidth usage at base stations, and periodically announces this information to

[2]We can only obtain current network loads for those networks under our control, see Section 6.2.1.

[3]meaning that we have code access to their base station,

Wireless Networks	Typical Bandwidth	Cell diameter	Charge Model	Charge Rate	Handoff Latency	Power Consumption
Infrared	1Mbps	7 meters	Free	0	2.3 s	349.6mW
WaveLAN	1.6Mbps	100 meters	Free	0	3.0 s	1148.6mW
Metricom	40kbps	Wide Area	Flat	$20.00 per month	9 s	External Battery
GSM Cellular	9.6kbps	Wide Area	per time	$0.40 per min	26 s	External Battery

Table 2: Network Characteristics in Our System

its coverage area. Since all data traffic goes through base stations, they have the most accurate information on current bandwidth usage, and the available bandwidth in the network. A network with base stations having higher available bandwidth likely offers better performance. Based on this information, we can design policies that achieve load balancing across different networks as shown in the next section.

However, we have to avoid the handoff synchronization of mobile hosts in the same vicinity. Several mobile hosts could discover the same better network and switch to it simultaneously, causing its load to increase dramatically and squandering its advantages. Almost immediately, the same mobile hosts will discover that the old network is now better, switching back together. The synchronization problem can cause instability for all these mobile hosts and poor performance. Its solution is addressed in Section 7.

Our scheme of estimating network conditions made the assumption that wireless links are always the bottleneck, but this is not always true. When a wired link is the bottleneck, we may be able to learn that from the SPAND performance server [11]. SPAND performance server determines network characteristics by making shared, passive measurements from a collection of hosts. By allowing base stations to obtain network conditions of the wired network from a SPAND server, base stations can in turn report that information to mobile hosts in its coverage as well. We will investigate this in the future.

Another drawback of our scheme is that it requires base station cooperation, thus eliminating operational WANs from our control. We will explore solutions to this in our future work.

6.3 A Policy Specification Model

6.3.1 Our Model

In this section, we present our policy specification model.

The cost of using a network n at a certain time is a function of several parameters: the bandwidth it can offer (B_n), the power consumption of using the network access device (P_n), and the cost (C_n) of this network [4].

$$Cost_n = f(B_n, P_n, C_n) \tag{1}$$

The bandwidth parameter estimates the current network condition. Power consumption and cost are parameters with fixed budgets; namely, mobile host's battery life, and maximum amount of money the user is willing spend for a period of time, respectively.

We can imagine that such a cost function is the sum of some normalized form of each parameter. Normalization is needed to ensure that the sum of the values in different units is meaningful. Users may specify the importance or weights of each parameter (i.e., w_b, w_p, w_c), which sum to 1. For those parameters that are not of concern to the user, she can set those weights to 0. Furthermore, weights may be modified by users or the system at run-time. This is especially important for parameters with fixed budgets, i.e., power consumption and cost. As the mobile host battery is running out or as the expenditure approaches the spending limit for a time period, w_p or w_c, respectively, should increase dramatically to reflect such a condition. The cost function of the network n, named as f_n, can be written as follows with $N(t)$ as the normalization function of parameter t:

$$f_n = w_b \cdot N(\frac{1}{B_n}) + w_p \cdot N(P_n) + w_c \cdot N(C_n) \tag{2}$$

$$\sum w_i = 1$$

Note that the lower the value of f_n, the lower the cost of network n is, and the better is network n. We take the reciprocal of the bandwidth B_n for this

[4] we only consider these parameters for our policy model prototype, other parameters can be easily added.

reason.

We turn to the normalization of these parameters. If a network offers twice as much bandwidth, but twice as expensive as the other network, then users consider these as equally good (they have the same cost function value). The property of logarithm $\log a - \log b = \log \frac{a}{b}$ can reflect this logic, and can also serve as normalization[5]. Therefore, we take logs for each factor:

$$f_n = w_b \cdot \ln \frac{1}{B_n} + w_p \cdot \ln P_n + w_c \cdot \ln C_n \tag{3}$$

$$\sum w_i = 1$$

If network n is not currently used, its available bandwidth B_n is $(B_{report} - Throughput)$ where B_{report} is the available bandwidth reported by network n's base stations, and $Throughput$ is the observed throughput at the mobile host.

When comparing two networks, their cost functions are calculated, and compared. The one with the lower value wins. This process is shown as follows:

$$f_1 = w_b \cdot \ln \frac{1}{B_1} + w_p \cdot \ln P_1 + w_c \cdot \ln C_1$$

$$f_2 = w_b \cdot \ln \frac{1}{B_2} + w_p \cdot \ln P_2 + w_c \cdot \ln C_2$$

$$f_1 - f_2 = w_b \cdot \ln \frac{B_2}{B_1} + w_p \cdot \ln \frac{P_1}{P_2} + w_c \cdot \ln \frac{C_1}{C_2}$$

If $f_1 - f_2$ is greater than zero, then network 1 is worse than network 2; if less, 1 better than 2; if equal, they are equally good.

We are using Formula 3 in our prototype. The power consumption and charge model and rate are static information stored in a network database object. The network bandwidth is dynamically computed. For those networks under our control (IR and WaveLAN in our system), we use the performance agent reporting scheme described in Section 6.2.1 to obtain the available bandwidth of the reachable networks. For commercial networks out of our control (Metricom Ricochet and GSM cellular modems), we use the "typical" bandwidth advertised by their vendors, as shown in Table 2.

[5]Note that the unit of a and b does not matter.

Now we illustrate the formula with some examples. Assume a mobile host has two reachable cellular networks (circuit switched) at this moment. One offers 19.2kbps charging 80 cents per minute, while the other offers 9.6kbps charging 40 cents per minute. Assume the policies only consider the cost and bandwidth, we have the following:

$$f_1 = w_b \cdot \ln \frac{1}{19.2kbps} + w_c \cdot \ln 80c/m$$

$$f_2 = w_b \cdot \ln \frac{1}{9.6kbps} + w_c \cdot \ln 40c/m$$

$$f_1 - f_2 = w_b \cdot \ln \frac{9.6}{19.2} + w_c \cdot \ln \frac{80}{40}$$

$$f_1 - f_2 = \ln 2 \cdot (w_c - w_b)$$

Therefore, if the user specified "cost is more important than bandwidth" ($w_c < w_b$), the network with 19.2kbps charging 80 cents/minute is better; if "cost is equally important as bandwidth", then both networks are equally good; if "cost is less important than bandwidth", then the network with 9.6kbps charging 40 cents/minute is better.

If a user wants to be connected to the cheapest network at all times, she should specify $w_c = 1$, and 0 for the other weights. In our testbed, that means: connect to the wireless LANs as much as possible; if not, connect to Metricom if possible (Flat rate); if none of the above available, connect to GSM cellular (40cents/minute).

If high performance is the most desirable to the user, she can assign $w_b = 1$, and the rest 0. This policy can achieve load balacing across different networks, and have the mobile host connected to the network with the highest available bandwidth. Again in our testbed, although IR typically offers higher bandwidth per coverage area than WaveLAN, the mobile host may choose to connect to WaveLAN because it is less loaded, and offers better bandwidth at that moment.

6.3.2 Stability Period

Periodically, the system re-calculates the cost function (f_n) of each reachable network based on up-to-date parameters. If a network is "consistently" better than the current network in use, the system hands off to the better network.

The word "consistently" is important. If a mobile user only transiently transfers to a better network, the gain from using the network may be diminished by the handoff overhead and short usage duration. On the other hand, networks like Infostations do offer very high bandwidth that may be available for short intervals. If the handoff latency to the Infostations is low enough, then even a very short usage duration is beneficial.

To determine whether incurring the cost of the handoff is worthwhile, we define *stability period* to be a waiting period before handoffs. Only if a network is consistently better than the current one in use for the *stability period* does the mobile host perform handoff. The stability period must be determined whenever the mobile host finds a handoff target.

We deduce the stability period as follows. First, we define T_{makeup} as the amount of time needed to make up the loss (i.e., loss of money or data depending on current policy) due to handoff latency $l_{handoff}$. If a mobile user is likely to be in the range of a better network for $T_{makeup} + l_{handoff}$ amount of time, then it is worthwhile to handoff. We define the stability period to be $T_{makeup} + l_{handoff}$ following the common practice of predicting the future from the recent past.

Now we derive T_{makeup}. If bandwidth is the only factor in consideration of the goodness of a network, let B_{better} and $B_{current}$ be the bandwidth of the better and current network respectively, and T_s be the *stability period*, then

$$(B_{better} - B_{current}) \cdot T_{makeup} = B_{current} \cdot l_{handoff}$$

Let

$$f = \frac{B_{better}}{B_{current}} \; f > 1$$

then

$$T_{makeup} = \frac{l_{handoff}}{f-1}$$

So,

$$T_s = l_{handoff} + \frac{l_{handoff}}{f-1} \tag{4}$$

Stability period (T_s) is defined as in Formula 4 when bandwidth is the only consideration. The fact that T_s depends on $l_{handoff}$ and f makes sense. When $l_{handoff}$ is short and the new network is many times better than the current network in use, the stability period is also short.

Now we generalize the calculation of T_{makeup} for all parameters of the cost function as follows:

$$T_{makeup} = \frac{l_{handoff}}{e^{f_{better} - f_{current}} - 1} \tag{5}$$

$$T_s = l_{handoff} + \frac{l_{handoff}}{e^{f_{better} - f_{current}} - 1} \tag{6}$$

f_{better} and $f_{current}$ are cost function values of the better and current network. The exponential term corresponds to f in our previous calculation (Formula 4).

7 Breaking the Handoff Synchronization

We mentioned the handoff synchronization problem in Section 6.2.1. Unlike other parameters in the cost function, the available bandwidth of a network depends on the number of mobile hosts it is serving and their bandwidth load, and it is highly dynamic. Handoff synchronization causes instability and poor performance for each mobile host.

We simulated a very simple scenaro to see the effect of handoff synchronization. Three mobile users share network access for a WaveLAN in-building network (1.2Mbps available for use) and a in-room IR LAN (1Mbps). Assume each of them has a fixed work load consuming 200kbps, 300kbps and 400kbps respectively. Their policies all specify $w_b = 1$; namely, bandwidth is the most important to them. We imagine that three of them walk into the in-room IR at the same time. The effect of handoff behavior on available network bandwidths is shown in Figure 2.

We can see the severe oscillation in this graph. The sharp troughs correspond to all three mobile hosts handing off to that network simultaneously causing dramatic load increase. Similarly, the peaks result from three of them leaving that network at the same moment. Observe that the peaks and troughs have a period of 6 seconds, which is exactly $l_{handoff} + T_s$. This is the cause of handoff synchronization.

We solve this problem through randomized stability period. A random number is generated, as the waiting period before handoffs, between the stability period defined in Formula 6 and five times of that value. Figure 3 results from two instances of the above scenario after applying this randomization.

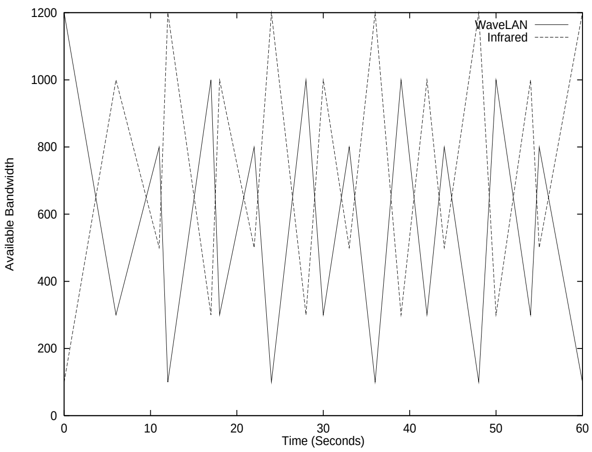

Figure 2: Handoff Synchronization

Both graphs show that the system stablizes after some initial handoffs. In the left graph, mobile host 1 first hands off to WaveLAN after 6 seconds, and mobile host 2 after 15 seconds, then the system stablizes from that point on. The right graph results from different randomized stability periods. In this case, mobile host 2 first switched to WaveLAN, then mobile host 3. This

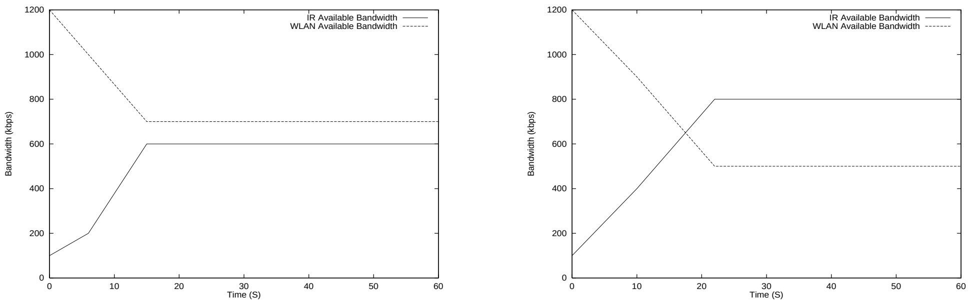

Figure 3: Break handoff synchronization through randomized stability period.

scheme essentially trades some additional waiting time before handoffs for the prevention of possible handoff synchronizations. However, is handoff synchronization a common phenomenon? If yes, additional waiting is worthwhile, and contributes to system stability. If not, this will only prevent users from utilizing networks efficiently. Noble, et al.[9] showed that when the transition to adapting to network conditions is not expedient enough, adaptation shows much less value. A more quantitative analysis of this tradeoff is our future work.

8 Implementation

In this section, we examine the implementation of our system. We first discuss the split object and event driven programming model in Section 8.1, then we present our software architecture in Section 8.2.

8.1 Programming Model

The *Split Object Model* [14] has been shown to be highly flexible and easily extensible. Because it uses the object oriented approach, code reuse is easily achieved. The software architecture in *split object programming model* splits into low overhead control functionality implemented in a scripting language (OTcl/Tk in our case) and performance critical data handling implemented in a compiled language like C or C++. Compiled objects provide core, composable mechanisms that are "glued" (or arranged and configured) through the scripting language to effect arbitrary application policies.

This programming model fits our needs exactly for a Policy-Enabled Handoff System. We implemented the raw handoff mechanism in C++, and light weight mechanisms, policies, graphical user interfaces in OTcl/Tk.

Inherent in OTcl/Tk is an event driven model. The burden of scheduling events or executions of a piece of script and event handler dispatching are all pushed to the Tk event loop. This greatly simplifies the program structure and implementation. In many places of our implementation, timers are used to schedule events or executions of a piece of scripts, such as periodically refreshing the GUI, taking samples of bandwidth usage, reporting the current bandwidth usage from the base stations, and conducting accounting. Without the event driven model, the implementation can be difficult and error prone.

8.2 Software Architecture

The Mobile Host object, as depicted in Figure 4, is the most complicated split object in our system. In C++, low level mechanisms such as the handoff operation and LAN reachability detection routines are implemented. On the

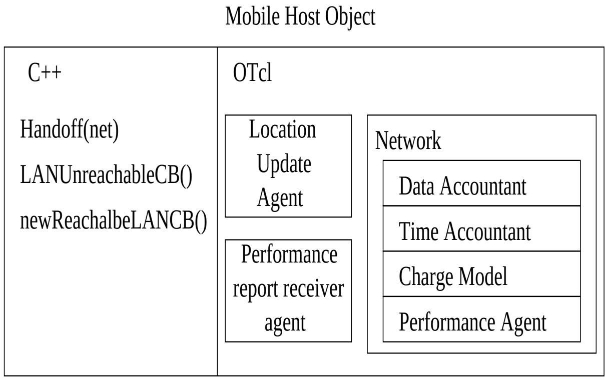

Figure 4: The Mobile Host Split Object

OTcl side, the mobile host object contains a set of network objects. For each subscribed network, there is a network object associated with it. The network object in turn contains a data accountant, time accountant, charge model, and performance agent object. The data accountant keeps track of the amount of data that has been transferred on this network. The time accountant keeps track of the amount of time the network has been used so far. The charge model object of the network records what charge model the network has (flat rate, free, per time or per bytes), and can calculate how much money has been spent for the network usage. Finally, the performance agent keeps track of the observed throughput of the mobile host using this network. In addition to the Network objects, the Mobile Host object also contains a location update agent which periodically updates the home agent on its current unicast care-of address when in a wireless WAN (Section 5). The performance report receiver agent is also part of the mobile host object responsible for receiving and processing the performance report from the base stations on available bandwidth of a reachable network.

Figure 5 gives the software architecture overview. The network DB object maintains properties for each network such as its typical bandwidth, latency, power consumption, charge model type, whether it is a LAN, whether it is connection oriented and its connection setup time if so. The Network object obtains this information by reading from an ASCII resource file (the content is basically what is in Table 2). Adding new networks is only a matter of adding a line to that file. The network DB object is referenced by almost all the objects in the system. The mobile Host needs information from the Network DB object to initialize its own Network objects.

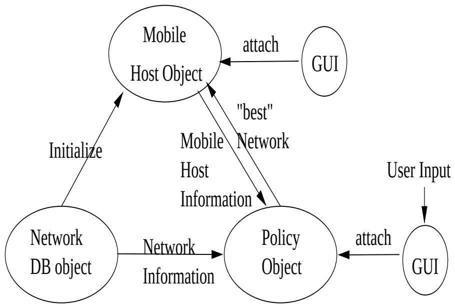

Figure 5: The Software Architecture Overview

The policy object is consulted by Mobile Host object periodically on what is the “best” network. The policy model described in Section 6.3 is implemented in our policy object. Other models can be implemented as well, as long as the policy object retains the same API. The policy object takes dynamic information from the Mobile Host (including all the information each Network object is keeping track of), and the static information from the Network DB object, and outputs the “best” network at the time.

Policy object can have a GUI object attached allowing users to enter their preferences, as shown in Figure 6.

Similarly, the Mobile Host object can be attached with a GUI object reflecting system status, allowing manual control of handoff operations, and loading user specified policy models. Figure 6 shows the user interfaces of our system. The traffic lights in the Network Manager window indicate the handoff status of a network. If it is black (WaveLAN in the figure), it means that the network is not reachable. If green (IR), it means that the network is in use. If yellow, it means that the mobile host is in the process of handing off to the network. If red, it indicates it is a reachable network not in use.

Initially, the system is in the manual mode. Users can click on the “Go” button to connect to a network. Automatic or policy mode can be invoked when users click on the “Load Policy” button. Users can load in their own policy module script. By default, the “standard.tcl” is loaded, which is our policy model (Section 6.3). “standard.tcl” brings up the Policy Specification window for users to specify weights for the parameters of the cost function. When users click on the “Show Info” button on the Network Manager win-

Figure 6: User Interfaces

dow, Network Information window pops up displaying current network and base station in use, the amount of money and time spent for this network so far, the amount of data transferred on it, current observed throughput, and base station bandwidth usage report.

9 Performance

The most ideal way to evaluate our system is to perform a user study, which we have not conducted at the time of writing. Through network usage traces in such a user study, we can observe how frequent handoff synchronization occurs. Then, we can evaluate the tradeoffs of using randomized stability period (Section 6.3.2). Also, a user survey would help us evaluate the user-friendliness of our policy model, and find an appropriate policy model if necessary.

In this section, we present the performance of our system from the aspect of handoff latencies. We experimented with four networks in our prototype: IBM Infrared LAN, Lucent WaveLAN, the Metricom Ricochet network, and the GSM cellular.

We define handoff latency to be the amount of time for a mobile host to handoff to a given network. It does not include the discovery time. We enabled the manual mode on our system, and measured handoff latencies from ten trials of manually triggered handoffs. The result is shown in Table 3.

We use reverse tunneling when mobile hosts are in wireless WANs, instead

Wireless Networks	Handoff Latency (sec)	Standard Deviation
Infrared	2.27	0.66
WaveLAN	3.01	0.26
Metricom Ricochet	9	1.1
GSM Cellular	26	3.9

Table 3: Handoff Latencies

of inserting mobile host's home address as source address into the outgoing packets, because firewalls will drop such packets as IP spoofing prevention. Through reverse tunneling, packets are routed to the home agent first, then to the correspondent host. We used "ping" to measure the worst case round trip time of reverse tunneling when we use Ricochet; namely, pinging another machine in the same subnet as the home agent. Without reverse tunneling, the round trip time between the mobile host and a correspondent host has an average of 0.60 seconds with a standard deviation of 0.51, while with reverse tunneling, it has an average of 0.70 seconds with a standard deviation of 0.33. Both of them have very high standard deviation. It is not clear why the case without reverse tunneling has much higher standard deviation. The round trip time for the reverse tunneling case is 17% higher than without on average.

10 Future Work

Hints can be considered in the policy model such as the moving speed of the vehicle, user activity history and pattern, and network coverage maps.

As described in Section 6.2, the current performance agent scheme has the drawback of base station cooperation and the assumption of the wireless link being the bottleneck. We need to explore other alternatives for more accurate performance reports. We also hope commercial networks can provide more information with regard to the global network condition within their networks.

Policy parameters described in Section 6.1 can interplay and co-relate. Reactions to the change of one parameter can cause changes to the rest of them.

In turn there are reactions to these changes, which can go on indefinitely. For example, high quality network service may impose high cost, but may reduce the connection duration, which in turn reduces the actual cost. Such chain effects can lead to instability. Also, user policies may contain conflicting requirements with regard to different parameters, and can halt the whole system if there is no special handling for such conflicts. Understanding the relationship between these parameters, and verifying user policies are part of our future work.

Our policy model does not take running applications into consideration. Application categories include bulk transfer (e.g., ftp), interactive (e.g., telnet), real-time (e.g., audio conferencing), and bandwidth intensive (e.g., video conferencing). Users specify priorities for each category and which applications belong to each. The weights for each parameter in the cost function can be per network and per application.

A refined policy model along these lines looks like this:

$$\begin{aligned} f_n = \sum_a (& w_{c,a} \cdot \ln C_{n,a} + w_{b,a} \cdot \ln 1/B_n \\ & + w_{p,a} \cdot \ln P_n + w_{f,a} \cdot \ln F_n) \cdot p_a \end{aligned}$$

For each application running at decision time, we calculate the cost function for it as before. Then, we multiply it by the priority of the application (the priority is converted to percentage here). Finally, we sum across all applications that are running at that time. We plan to incorporate this enhanced policy model into our system.

11 Conclusion

In this report, we have presented a *policy-enabled handoff system* across heterogenous wireless networks. Policy enabling greatly improves the system flexibility and extensibility. It allows users to issue policies and have their mobile host connected to the most desireable network to them. We described a network condition estimation scheme along with other policy enabling mechanisms. We also experimented with a policy model. We believe we have achieved the principle of *user involvement with minimal user interactions*. We also addressed the issue of system stability. We used *stability period* to ensure a handoff is worthwhile. We also identified the problem of handoff synchronization, and solved the problem through randomized stability period. We demonstrated that a stablized policy-enabled handoff system can achieve load balancing, and improve network performance.

Our experience of using the *split object model* re-enforced its merit of flexible glueing of mechanisms together using scripting language. The separation of policy and mechanism is a natural outcome of such a programming model.

12 Acknowledgement

I would like to thank Professor Randy Katz and Anthony Joseph for their insightful ideas and comments all along. I would also like thank Mark Stemm for helping us setting up and running his vertical handoff system. We thank the following people for the lively discussions: Adam M. Costello, Tom Henderson, Giao Nguyen. We also thank the researchers from DaimlerBenz for their input. This project is funded by DARPA and MICRO.

References

[1] Jian Cai and David J. Goodman. General packet radio service in gsm. In *IEEE Communications Magazine*, October 1997.

[2] Stuart Cheshire and Mary Baker. Internet mobility 4x4. In *ACM SIGCOMM August,1996*, August 1996.

[3] Daedalus wireless research group. http://daedalus.cs.berkeley.edu/.

[4] D.J. Goodman, J. Borras, N.B. Mandayam, and R.D. Yates. Infostations : A new system model for data and messaging services. In *Proceedings of IEEE VTC'97*, pages 969–973, May 1997.

[5] Index project. http://www.index.berkeley.edu/.

[6] David B. Johnson. Scalable support for transparent mobile host internetworking. In *Wireless Networks*, 1996. Revised from a paper presented at the Ninth Annual IEEE workshop on Computer Communications, IEEE Communications Society, 1994.

[7] Anthony D. Joseph, B. R. Badrinath, and Randy H. Katz. The case for services over cascaded networks. In *The First ACM/IEEE International Conference on Wireless and Mobile Multimedia (WoWMoM'98)*, 1998.

[8] Randy H. Katz and Eric A. Brewer. The case for wireless overlay networks. In *Proceedings of SPIE Multimedia and Networking Conference*, January 1996. San Jose, CA.

[9] Brian D. Noble and et al. Agile application-aware adaptation for mobility. In *Proceedings of SOSP-16*. ACM, October 1997. Saint-Malo, France.

[10] Srinivasan Seshan, Hari Balakrishman, and Randy H. Katz. Handoffs in cellular wireless networks: The daedalus implementation and experience. In *Kluwer International Journal on Wireless Communication Systems, 1996*, 1996.

[11] Srinivasan Seshan, Mark Stem, and Randy H. Katz. Spand: Shared passive network performance discovery. In *Proc 1st Usenix Symposium on Internet Technologies and Systems*, December 1997.

[12] Spaceway. http://www.hcisat.com/.

[13] Mark Stemm and Randy H. Katz. Vertical handoffs in wireless overlay networks. In *ACM Mobile Networking(MONET), Special Issu on Mobile Networking in the internet*, 1997.

[14] Eric Brewer Steven McCanne, Randy Katz, Lawrence Rowe, Elan Amire, Yatin Chawathe, Ketan Mayer-Patel Alan Coopersmith, Suchitra Raman, Angela Schuett, David Simpson, Andrew Swan, Teck-Lee Tung, David Wu, and Brian Smith. Toward a common infrastructure for multimedia-networking middleware. In *7th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, May 1997.

[15] Teledesic. http://www.teledesic.com.

[16] Wideband cdma, 3rd generation technology. http://cebit.ericsson.se/wcdma/wcdma/.

[17] Xinhua Zhao and et al. Flexible network support for mobility. October 1998.

A Object APIs

This appendix describes the APIs of important objects in our system.

A.1 MobileHost Split Object

Mobile Host object is a split object which has its member functions implemented both in C++ and OTcl.

In OTcl, the following member functions are defined:

- init (homeAddr, homeAgentAddr): constructor of the MobileHost given the mobile host's home IP address and its home agent's IP address.
- enableReverseTunnel(): enables the reverse tunneling; namely, the packets sent from the mobile host are routed to its home agent first through IP-in-IP encapsulation, then the home agent forwards it to the correspondent host. This is generally needed for wireless WANs because of the deployment of the firewalls.
- disableReverseTunnel(): disables the reverse tunneling. This is to take the advantage of the case when no firewall is deployed, or the communication is within the same subnet.
- setManualMode(): puts the system in the manual mode so that users decide when to handoff by clicking on the "Go" button on the MobileHost UI.
- resetManualMode(): puts the system in the automatic mode. And the policy is used to decide when to handoff.
- getSwitchableNetTypes (): gets the reachable networks.
- getAllNetworkTypes (): gets all the configured networks.
- getAllLANs(): gets all the LANs.
- getAllWANs(): gets all the WANs.
- setCurrentNet(): sets the current network in use.
- getTotalCharges(): gets the total expenses so far.
- getTotalChargesWithoutFlat: gets the total expenses so far without considering the Flat rate networks.

- getCurrentNetType (): gets the current network in use.
- getCurrentBS (): gets the current base station in use.
- recvCB(): callback when receive a UDP packet (performance report in our case).
- setPolicyObj (policyObj): associates a policy object with this mobile host.
- getNetObj(netType): gets the network object given its type.
- candNetTimeout(): callback after the mobile host has waited the stability period before handing off to a candidate network.
- analyzeState (): this function is called periodically to analyze the current network condition at automatic mode (when using a policy to drive handoffs).
- handoffTo (netType): performs handoff to a given network.
- connectToWAN (wanType): sets up a connection to a WAN, but not actually use this network yet.
- invokeWANUsage (wanType): sets up default router to use the WAN that has already been connected.
- disconnectFromWAN (wanType): tears down the connection with a WAN.
- startAnalyzeState (): starts to analyze the current network condition periodically; starts the automatic handoff mode.
- stopAnalyzeState (): stops the automatic handoff mode, and starts the manual handoff mode.
- handoffCB (): callback after a handoff, should be called after each handoff.
- netUnreachableCB (): callback called from C++ side after detecting some networks are not reachable any more.
- newReachableNetCB (): callback called from C++ side after detecting newly reachable network.

In C++, the following member functions are defined:

- handoffToWANCB (): callback after handoff to a WAN, should be called after handing off to a WAN.
- clearOld (): updates the base station reachability status, and removes out-of-reach base station from our list of reachable base stations and networks.
- setPolicy (): records the policy object name at C++ side.
- doHandoff (network, BSAddr): performs handoff to a LAN given which network to handoff to, and the base station IP address to use.
- getBSforNet(network): get the base station for a given network.

A.2 Network Object

Network object maintains both the static and dynamic information about a network.

- setCareof(careof): sets the mobile host's care-of address in this network.
- setGateway (gateway): sets the the gateway in this network that serves the mobile host.
- setIfname (ifname): sets the interface name of network.
- setBSBWUsage (bw): sets the base station bandwidth usage for this network.
- getNetworkType (): gets the network type.
- getIfname (): gets the interface name used by this network.
- getCareof (): gets the mobile host's care-of address.
- getGateway (): gets the gateway that is serving the mobile host.
- getCharges (): gets the current expenses caused by using this network.
- startAccounting (): starts periodic accounting for this network.
- stopAccounting (): stops accounting.
- getPerceivedBW (): gets the perceived bandwidth from this network.

- getTimeDuration (): gets the time duration this network has been used..
- getBytesTransferred (): gets the amount of data that have been transferred on this network.
- getBSBWUsage (): gets the base station bandwidth usage at this network.

A.3 Policy Object

At automatic handoff mode, the policy object is referenced by "analyzeState" function of MobileHost object for the best network.

- getBestNet (): gets the best network.
- disconnectWAN(wanType): disconnects a WAN, this is a policy decision because one needs to decide whether to tear down the connection completely, or to keep the connection, and only manipulates the routing table.

A.4 Accountant Object

Accountant object performs the task of accounting for the sytem. DataAccountant and TimeAccountant are subclassed from Accountant object accounting data and time. These objects are used by performance agent to obtain the throughput on the mobile host and base stations.

- init (interval): contructor that takes the argument "interval" indicating the frequency of accounting activity.
- start: starts accounting.
- stop: stops accounting.
- doAccounting: performs accounting.
- get: gets accounting information. If it is DataAccountant, it gets the amount of data that have been transferred over a network interface; and if it is a TimeAccountant, it gets the amount of time using a network.

A.5 PerfAgent Object

PerfAgent keeps track of the throughput on a network interface. It uses both DataAccountant and TimeAccountant objects.

- init (numSamples, interval, ifname): the constructor. "interval" indicates the frequency of taking the measurement for throughput. "numSamples" is the number of samples or the number of intervals the average throughput is taken over.
- start (): start running the performance agent.
- stop (): stop running the performance agent.
- getEstBW (): get the estimated bandwidth (throughput).

A.6 ChargeModel Object

ChargeModel object models different charging models. It is subclassed to ChargeModel/Free for free networks such as WaveLAN and Infrared, ChargeModel/Flat for flat rate networks like Metricom, ChargeModel/Time for networks that charge per time like GSM cellular modem, and ChargeModel/Data for networks that charge per data amount.

- init (netobj): contructor that takes argument of a network object.
- getCharges (): gets the charges.
- getModelType (): gets the model type.

A.7 KernelAgent Split Object

KernelAgent object is an object that interacts with the kernel. It sets IP options through OTcl commands. It is a split object with all the member functions implemented in C++.

- setsockopt (option, data): sets the IP socket option given the option and data to set in the kernel.

A.8 UDP Split Objects

Some UDP utility objects are implemented to send and receive UDP packets. UDPSender object sends UDP packets, and UDPReceiver object receives UDP packets. They are split objects with all the member functions implemented in C++.

- UDPSender object:
  - init (IPAddr, port): constructor that takes the destination IP address and port as arguments.
  - send (data): sends data in a UDP packet.
- UDPReceiver object:
  - init(port): constructor that takes port listening on as argument.
  - setHandlerObjName (handlerObj): sets the handler object. The handler object must implement a function called "recvCB".