DESIGN AND IMPLEMENTATION OF AN INTER-CELL MANAGEMENT SYSTEM

THE SABINO SYSTEM

by

Fahd K. Al-Bin-Ali

A Thesis Submitted to the Faculty of the

DEPARTMENT OF COMPUTER SCIENCE

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

WITH A MAJOR IN COMPUTER SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

2 0 0 2

ABSTRACT

Wireless networks based on IEEE 802.11 are becoming increasingly widely deployed. However, 802.11 systems fail to make optimal use of the available network resources because of a lack of coordination between different system components, in particular, the absence of any form of coordination between access points and mobile nodes. Consequently, wireless network installations may not make optimal allocation of mobile nodes to access points, leading to imbalances in the load distribution and potentially reducing the network bandwidth available to users. This thesis presents the Sabino system: a lightweight management architecture that coordinates the actions of access points and mobile nodes to ensure, where possible, a more balanced load distribution and hence better utilization of system resources. Sabino as an architectural solution is applicable for many other applications and future wireless deployments in which mobile nodes have different Quality-of-Service requirements and are serviced by different network technologies. Key aspects of the Sabino system include easy and incremental deployment, flexible architectural topology, minimal overhead and effective management of resources. This thesis presents evidence of some of the problems in existing 802.11 systems and it describes the design, the implementation and the evaluation of the Sabino system as an effective solution to these problems.

ACKNOWLEDGMENTS

I express my deepest thanks to God for giving me the strength and the knowledge to finish this work. I thank my mother, my father and my family for their guidance and support throughout my life.

I would also like to express my gratitude to my advisor, Dr. Nigel Davies, for giving me the opportunity to work with him. His innovative ideas and continuous guidance and support have been invaluable. I thank Prasad Boddupalli for his enormous efforts during the making of this work. I also thank Dr. Alon Efrat for his valuable insights.

I would also like to thank the Department of Computer Science at the University of Arizona. I thank all lab staff members, including John Luiten, Phil Kaslo, Sandy Miller and John Cropper.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past decade there has been a rapid growth in the popularity of *Wireless Local Area Networks* (WLANs). Like their wired counterparts, WLANs are designed to provide high bandwidth to users in limited geographic areas. However, WLANs allow users to maintain connectivity as they move without being tethered by the constraints of physical connections. Schools and universities are installing WLANs to bridge students to the Internet [Hills, 96], [Comer, 95]. Malls are providing customers with wireless access allowing them to retrieve prices, search shops and purchase products [Bahl, 00]. Tourist sites are providing wireless devices to aid tourists in navigating areas, exploring sites and learning about attractions [Friday, 01]. Moreover, with the deployment of Fourth Generation (4G) networks, WLANs are expected to be an integral component of the public wireless network infrastructure [Nakajima, 01].

Embraced by many vendors, IEEE 802.11 has become the de-facto wireless LAN standard. The benefits of standardization coupled with the low cost of deploying 802.11 based networks have made it a popular solution for providing wireless connectivity. However, 802.11 systems face numerous challenges, for example, the increase in bandwidth demanding applications requires efficient allocation and sharing of network resources between mobile hosts. These challenges have resulted in a significant body of research aimed at improving the performance of 802.11 networks. Such research has studied, for example, Quality-of-Service (QoS) [Deng, 99], [Kopsel, 01], resource allocation [Pradhan, 98], handoff management [Caceres, 96] and various Medium Access Control (MAC) enhancements [Weinmiller, 96]. These efforts have typically focused on improvements within a single cell and have required changes to the MAC architecture. However, one of the unique characteristics of wireless systems is that it is possible to have multiple cells covering a single geographic area. In fact, it is possible to have multiple networks covering the same geographic area creating so-called *overlay networks* [Katz, 96]. This results in aggregating the bandwidth in these overlapping coverage areas. In previous research it has been shown that handoffs can be achieved between vertical network overlays [Stemm, 97], [Wang, 99] to improve the end-end communication quality available to mobile nodes. Using overlapping cells belonging to a single network is not without problems: applications must tolerate the changes in bandwidth as mobile nodes roam across different cells that might be subject to different loads or might support lower bandwidths. Existing 802.11 systems fail to make optimal use of the resources available in such configurations due to the lack of coordination between various system components. In particular, current installations lack any kind of coordination between overlapping access points and mobile nodes, leading to imbalances in the load distribution and potentially wasting unused network resources.

This thesis describes a novel architectural solution to solve this problem: the Sabino system. Sabino is a suite of software designed to run on (or close) to 802.11 access points and mobile nodes. It provides a coordination framework to manage network resources using the underlying supported network primitives such as Simple Network Management Protocol (SNMP) and Management Information Base (MIB) objects [McCloghrie, 91]. Sabino coordinates the association and dissociation of nodes to access points, therefore performing better load balancing, which results in an overall improved system performance. Moreover, Sabino as an architectural solution is applicable for future wireless deployments in which overlapping cells might use different MAC protocols targeted at different traffic types. In such cases, Sabino can manage the network resources using the appropriate primitives supported by each network technology to provide improved overall utilization of resources. Finally, it is worth noting that unlike numerous research efforts, the architecture presented does not require any changes to the MAC protocol, which makes it easily deployable.

The remainder of this thesis is structured as follows. Chapter 2 provides a brief overview of the IEEE 802.11 standard, focusing on its mechanisms for creating associations between mobile nodes and access points, the roaming mechanism it uses and how Agere Orinoco 802.11 products implement load balancing and the technical flaws in their scheme. Chapter 3 contains a detailed description of the Sabino system including the different topological architectures possible, the interaction interfaces required and detailed interaction scenarios that illustrate the operation of the system. Chapter 4 presents a discussion of the implementation of the different components of Sabino and the experimental setups used to conduct our experiments. Chapter 5 presents the results of these experiments and a detailed analysis and evaluation of the implementation. Finally, chapter 6 discusses plans for future work and some concluding remarks.

Chapter 2

Overview of 802.11

The IEEE 802.11 standard [IEEE, 99a] was designed to specify a cheap, robust wireless networking technology. The standard provides two modes of network configuration: ad-hoc and infrastructure. The former provides a means to create a group of mobile nodes without any infrastructure support, while the latter is typically used to provide continuous network coverage in certain geographic areas using base stations known as access points. 802.11 supports 14 partially overlapping channels of which 3 channels do not overlap, allowing network designers to aggregate the bandwidth within highly loaded geographic areas by overlapping access points using non-interfering channels. The standard specifies two distinct MAC protocols, namely *Distributed Coordination Function* (DCF) and *Point Coordination Function* (PCF), that offer contention based medium access and contention-free medium access respectively [IEEE, 99a].

The IEEE 802.11 workgroup has extended the original standard by introducing the 802.11b specifications [IEEE, 99b] for high data rate support including two new speeds 5.5 Mbps and 11 Mbps. In addition, a task group known as 802.11e [IEEE, 01b] has been formed to enhance the current 802.11 MAC protocols and to expand support for applications with QoS requirements. The QoS baseline proposal [Godfrey, 01] contains two different access methods and three QoS levels to accommodate the needs of *Integrated Services* and *Differentiated Services*. The former provides end-to-end connection-oriented QoS with support for streaming and centralized scheduling, while the latter provides a simple mechanism for prioritizing traffic within each cell.

2.1 Association in 802.11

Typically, 802.11 systems are configured with multiple access points, of which, some overlap to provide higher aggregate bandwidth within heavily loaded geographic areas. 802.11 supports the association/re-association/dissociation of mobile units with access points to enable roaming. Each mobile node is equipped with a wireless adapter that implements the roaming algorithm. Creating the initial association of a mobile node with an access point starts with a process called scanning. The IEEE 802.11 standard defines two methods of scanning: *passive scanning*, where the station switches to a channel and listens to beacons from access points that use that channel and *active scanning*, where the station switches to a channel and issues a so-called *Probe Request*, to which a *Probe Response* is expected within a given time frame. Most 802.11 product vendors implement active scanning as it provides a faster and a more efficient mechanism for detecting the access points in the vicinity of the mobile nodes. Performing a series of scans on different frequencies is called *sweeping*. There are two types of sweeps: *full sweeping* which goes through all the channel-list and *short sweeping* which skips the channels that do not have

sufficient frequency distance from known active channels [Lucent, 98b]. Figure 1, illustrates a scenario with a single mobile node trying to associate with one of two overlapping access points. The mobile node is in the overlapping coverage area of AP1 and AP2, it associates with AP1 as it has a stronger *Signal-to-Noise Ratio* (SNR).

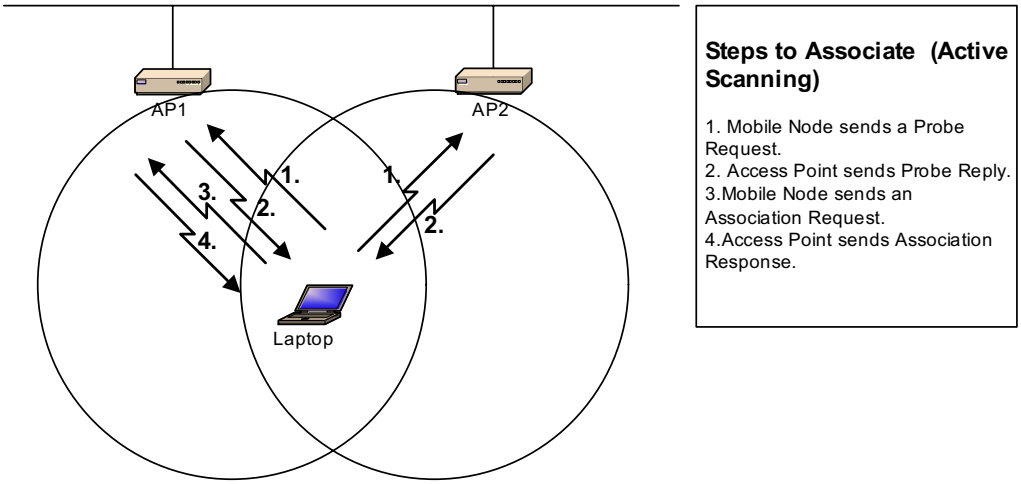

Figure 1: Association in 802.11 Networks

Information about available access points is stored in the wireless adaptor card. After updating the access point list stored in the card, the mobile node sends an *Association Request* to the access point with the strongest signal strength (AP1). Assuming that the mobile node is not denied access, the access point responds with an *Association Response* that in effect binds the mobile unit to the access point.

While having multiple overlapping cells is typical in 802.11 networks, mobile nodes can only re-associate with new access points in two situations: during roaming or for load balancing purposes. The following sections describe in detail these two features.

2.2 Roaming in 802.11

When a mobile unit moves away from the access point, the SNR of the link drops, and will eventually drop below a threshold value known as the *cell search threshold*. When this event occurs, it triggers the roaming algorithm to start looking for other access points to associate with. In this process, the mobile unit initiates a *sweeping* that constructs an updated access point list. When the SNR drops below a second threshold known as *cell switching threshold* and defined as *'cell search threshold – Delta SNR'*, the roaming algorithm triggers a re-association by selecting another access point from the access point list and it issues an association request to it as described earlier. Figure 2 shows the relationship of the SNR of two access points as a mobile node roams from AP1 to AP2.

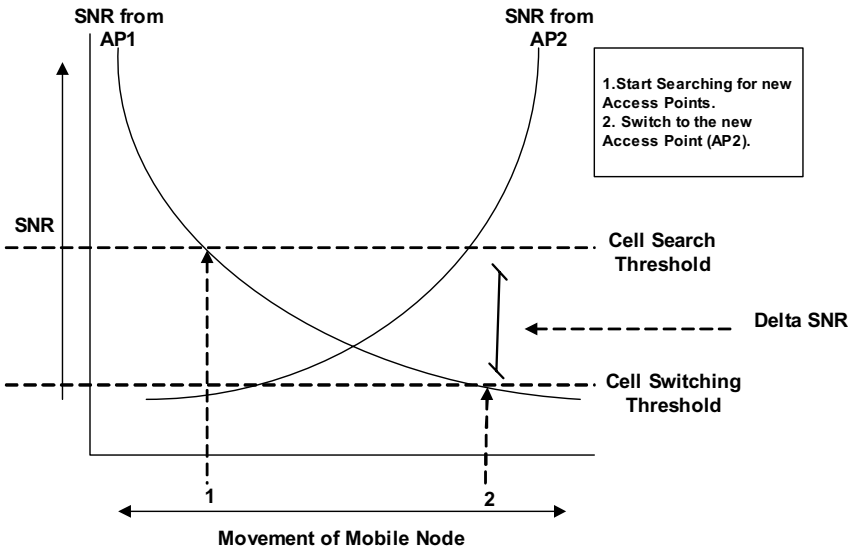

Figure 2: Roaming and SNR in 802.11

In most commercial implementations of 802.11, it is possible to associate a network name (SSID) with an access point. Mobile nodes can choose to associate with any access point or to restrict their selection by providing a network name that must match the access point they want to associate with. In this way, it is possible to create multiple logical networks within a given geographic area by using different network names for different sets of access points. One caveat is that mobile nodes do not roam between logical networks. Moreover, even if a node does not specify a network name, once an association has been made to a named access point the node will only roam to access points with this same network name.

Finally, it is worth noting that manufacturers such as Agere allow a single physical access point to have two separate wireless network interfaces, each with a logical network name (the network names may be the same or different depending on the configuration). This is essentially an optimization and can be thought of as two logically separate access points that happen to be co-located. For example, it is possible for a mobile node to roam between two different network interfaces provided by a single access point (assuming they have the same logical network name). In this thesis when access points are discussed, we are assuming a logical view unless otherwise stated.

At present there is no standard for inter-access point coordination. However, an IEEE task group known as 802.11f has been formed to design a protocol called *Inter Access Point Protocol* (IAPP) [IEEE, 01a]. This protocol will include hand-over procedures to update the learning bridges on different access points and to support the interoperability of different 802.11 products.

2.3 Load balancing in 802.11 Products

Many 802.11 systems [Lucent, 98a] implement a basic load balancing mechanism, for example, Agere Orinoco 802.11 wireless products trigger a load-balancing algorithm

once 4 consecutive beacons are lost by a mobile node. Beacons are special packets that are used to maintain synchronization between mobile nodes and access points. They are sent at nominal rates by 802.11 access points, for example, Orinoco AP-1000 access points issue a beacon every 100 milliseconds. It is assumed by Orinoco products that losing 4 consecutive beacons is an indication of contention in accessing the shared medium, as a result, the load balancing algorithm gets triggered to search for an alternative overlapping access point to associate with. Fundamental to examining the validity of this assumption is to understand the reasons that cause beacon loss, and this requires analyzing how 802.11 resolves contention to the shared medium.

There are two factors that determine who seizes the shared medium in 802.11: a static element represented by a fixed *Inter-Frame Space* (IFS) and a random backoff interval. Figure 3, shows the relationship of these intervals as mobile units send their frames.

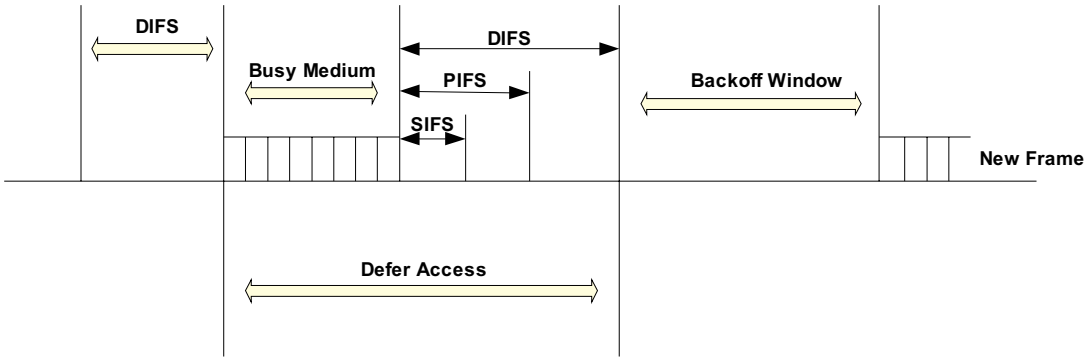

Figure 3: Medium Access in 802.11

IFSs provide fixed priority levels for accessing the wireless media. The standard defines four IFSs in ascending order of time: *Shortest IFS* (SIFS), *PCF IFS* (PIFS), *DCF IFS* (DIFS) and *Extended IFS* (EIFS). Control packets such as acknowledgments use SIFS, while data packets and management packets (such as beacons) use DIFS. Therefore, in an event of a contention, acknowledgements will have higher priority than data and management packets. It is worth noting that the IFSs are fixed values determined by the physical layer.

The random backoff scheme in 802.11 is designed to minimize collisions when they are likely to occur, in particular, when the medium is idle. The backoff algorithm gets triggered in 2 situations: when either *physical carrier sensing* or *virtual carrier sensing* fail. Physical carrier sensing refers to the detection of a collision through listening to the shared medium, while virtual carrier sensing refers to the detection of an ongoing Request-To-Send/Clear-To-Send (RTS/CTS) soft reservation by a contending mobile node. All packets including beacons are subject to backoff delays due to contention like any other packets, however, access points delay all data and control packets until delayed beacons get transmitted. 802.11 uses two different ways for computing the random backoff for beacons and data packets. The following equations define how the computation is done:

$$Data \quad Backoff \quad Time = random(0, CW) \times aSlotTime$$

$$Beacon \quad Backoff \quad Time = random(0, 2 \times CW_{\min}) \times aSlotTime$$

random(min,max) is a function that generates a pseudorandom number using a uniform distribution from the range [min,max] inclusive. aSlotTime is a value determined by the physical properties of the shared medium. CW can take any value between CWmin=7 and CWmax=255 in a predefined exponentially increasing series. Possible values are defined by the following list (7,15,31,63,127,255). Initially when the station starts up, CW is initialized to CWmin. If collisions start to occur, CW takes the next higher value in the series, therefore expanding the contention window from which a random value is generated and in effect reducing the probability of colliding with a competing mobile node [IEEE, 99a].

From the above equations, we draw some intuitive observations that explain why Orinoco AP-1000 fails to achieve proper load balancing, in particular, we explain why losing 4 consecutive beacons is ineffective as an indicator of contention in accessing the shared medium. AP-1000 access points (equipped with Orinoco Silver cards) can support a bandwidth of 5.5 Mbps and can transmit packets of size 1500 bytes. Therefore, the transmission time for one packet is computed using the following equation:

$$Packet \quad Transmission \quad Time = \frac{Packet \ \ Size}{Data \ \ Rate}$$

$$= \frac{1500 \times 8}{5.5 \times 10^6}$$

$$= 2.2 \ \ m\sec$$

In addition, AP-1000 sends a beacon every 100 milliseconds, therefore in order to miss 4 consecutive beacons, the medium must stay busy for at least 300 milliseconds. The following equation computes how many transmissions are required to occupy that amount of time:

$$Number \quad of \quad Transmissions = \frac{Total \quad Transmission \quad Time}{Packet \quad Transmission \quad Time}$$

$$= \frac{300}{2.2}$$

$$\approx 136 \quad Transmissions$$

Therefore, at least 136 consecutive transmission attempts are needed (prior to having a successful beacon transmission) in order to trigger the load balancing algorithm. These transmissions can succeed or can fail due to collisions. It is important to emphasize that mobile nodes cannot seize the shared medium more than once in an event of having a delayed beacon at the access point (unless their retransmission timer times out), this is a result of the access point delaying the acknowledgments (which takes place if a beacon gets delayed).

Given the previous observations, assume there is a single mobile node in the coverage area, it is impossible for it to seize the medium for 136 consecutive transmissions for the simple reason that the access point will delay the acknowledgments destined to the node in an event of having a delayed beacon, therefore the beacon will be delayed by at most 1 preceding transmission. However, having an additional mobile node makes it possible to seize the medium for 136 consecutive transmissions, this can occur if the 2 mobile nodes start transmitting simultaneously, and their packets collide with each other for 136 consecutive transmissions and the backoff random number that they select from their contention window is always smaller than the backoff random number chosen by the access point for the beacon backoff time. Notice that each collision will cause an expansion to the contention window, therefore the probability of selecting the same backoff time for both mobile nodes will decrease as collisions occur and the probability of the random number being smaller than the random number chosen for the beacon backoff time will also decrease. It is clear that the probability of this happening for 136 consecutive times is extremely low. In addition, as the number of mobile nodes in the coverage area increases, the probability of collisions occurring increases leading to an increase in the size of the contention windows, therefore increasing the probability that beacons will succeed as the range of the contention window for mobile nodes will become much larger than the range of selections for beacons. It is important to emphasize that providing a probability model for beacon loss is out of the scope of this thesis, however, it is intuitive that the backoff mechanism used by 802.11 does not permit the loss of 4 consecutive beacons easily. Finally, IEEE 802.11 does not suggest any mechanism for load balancing across multiple access points, products such as Agere Orinoco AP-1000 have devised their own mechanisms.

The next chapter presents a detailed design of Sabino. Sabino overcomes the limitations that exist in the load balancing scheme used by Agere Orinoco 802.11 products. This is achieved by effectively coordinating the association and dissociation of mobile nodes to access points. However, this is one application of Sabino. In fact, Sabino is a management framework that can be used to optimize many different attributes, one of which is load balancing.

Chapter 3

Design of The SABINO System

3.1 Deployment Scenario

Figure 4 illustrates a typical deployment scenario of an 802.11 based network in public access environments. Three access points (AP1, AP2 and AP3) are being used to provide coverage of a certain geographic area. AP1 and AP2 provide almost complete coverage of the area required and AP3 has been added to offer increased throughput for the most heavily loaded area and to fill any coverage holes left by AP1 and AP2.

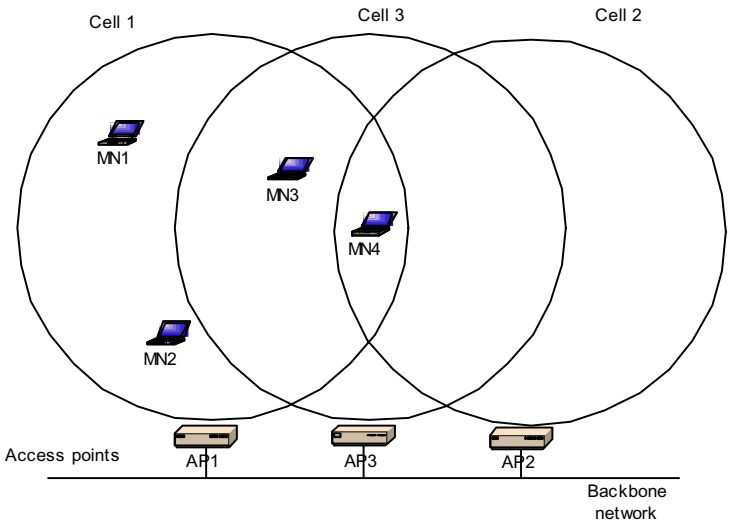

Figure 4: Example Wireless Network Installation

Given an initial configuration of the system in which several mobile nodes (MN1, MN2, MN3 and MN4) are connected to AP1 and none are connected to AP2 or AP3, as the load in AP1 increases, the probability of each of the mobile nodes seizing the shared medium decreases and the probability of having collisions increases as the mobile nodes attempt transmitting their packets simultaneously. Therefore, the overall throughput of the system will decrease since the mobile nodes are contending for only the channel provided by AP1. It is clear that the overall throughput of the system could be increased if one or more mobile nodes moved from AP1 to another overlapping access point to reduce the contention in cell 1, for example, AP1 and AP3 are visible for MN3, therefore, if MN3 decides to move to AP3 the load on AP1 will decrease. Also, AP1, AP2 and AP3 are visible to MN4, therefore if MN4 decides to move to AP2 or AP3 the load on AP1 will decrease as well. However, in current systems this form of load balancing does not

take place. The experiments, described in detail in chapter 5, confirm that even products such as Agere's Orinoco 802.11 (that encompass elementary load balancing mechanisms) do not prove effective at supporting that form of system optimization. In more detail, the Agere system enhances the basic 802.11 roaming algorithm by introducing an implicit load-balancing scheme that triggers a re-association if 4 consecutive beacons transmitted by an access point are missed by the mobile node. Of course such a scheme is unable to distinguish between bursts of interference and excess load in a cell. Moreover, by means of simple experimentation we were able to demonstrate that such a simplistic load balancing approach does not perform well in practice. We established a test network of two overlapping cells provided by a single Orinoco access point configured with two wireless cards. This network was used to support two mobile nodes that initially connected to the same wireless interface on the access point. As we increased the load generated by these nodes the system failed to cause either of the mobile nodes to effect a handoff to the alternative wireless interface that was providing an overlapping network cell. As a consequence, the overall throughput of the system was significantly reduced when compared to a system in which the mobile nodes connected to different wireless interfaces on the access point. The intuitive explanation for this failure in effecting a handoff is explained in section 2.3.

In the following sections we describe in detail the architectural design of the Sabino system. Sabino is a suite of software components designed to run on access points and mobile nodes, it provides a mechanism for effective load balancing to address the problems described above. It is important to emphasize that load balancing is one parameter that Sabino can optimize. In fact, Sabino provides a management framework that enhances the coordination between various system components to improve the overall performance of the network.

3.2 Architectural Topologies

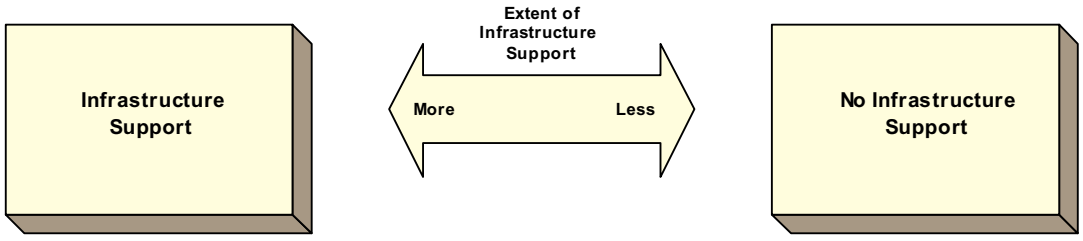

Figure 5: Architectural Topologies for Sabino

One key design decision is the extent of infrastructure support needed to run Sabino effectively. Figure 5 illustrates the spectrum of topological choices for Sabino, it extends from having an architecture with no infrastructure support to having one with infrastructure support only. Running Sabino on the infrastructure with no support on the mobile nodes is an attractive solution, particularly, because of the transparency of this approach since mobile nodes will not be required to run any additional software. However, the collaboration between different system components decreases and the architecture becomes more centralized which creates some limitations. Alternatively, the

other end of the spectrum suggests an architectural solution that runs with no infrastructure support, therefore, only mobile nodes will be required to run Sabino. This solution might be more economic since no additional components will be added to the infrastructure. However, the overhead of the interaction between the distributed mobile nodes will be greater, therefore reducing the shared bandwidth. In addition, the interaction protocol between the mobile nodes is likely to be more complex than a more central one (with some infrastructure support). In this section, we explore three architectural choices, an infrastructure based topology that collects all the load information in a single node on the infrastructure and requires no additional assistance from any of the mobile nodes in the coverage area, a no-infrastructure based topology that runs with no infrastructure support and requires the collaboration of mobile nodes, and finally, a hybrid topology that requires both infrastructure support and mobile node support.

3.2.1 Infrastructure Topology

This is the simplest architectural choice, it is based on client-sever interaction, with a single station (an access point or a dedicated machine) on the infrastructure managing a number of access points and keeping track of the load information on each one of them. If the load increases above a specific threshold, the station selects a node to move from one access point to another overlapping one. We describe the main component in this topology, namely, the Network Manager.

3.2.1.1 The Network Manager

The central component to this topology is the Network Manager: a station (access point or a dedicated machine) that is designated as the manager of several access points. It runs a management module that stores a list of all the access points on the network and has privileges to access each one of them. The Network Manger can modify the configuration of any access point in its list, this includes specifying which mobile nodes get to connect to which access points. However, reconfiguring an access point is an expensive operation that requires restarting the access point, in particular, the access point must reload its kernel. The Network Manager is also responsible for keeping track of the load information on each access point in the system. Our system is agnostic with respect to the method used for extracting this kind of information from the access points. For example, we could run the manager on a gateway to which all access points are connected, and simultaneously, run on it a monitoring module to infer load information, another possible way for extracting such information is through using special management interfaces such as SNMP. Deciding on a specific strategy for extracting this information is dependant on the protocols supported by the access points and any available vendor specific APIs.

It is clear that this topology gathers all load information required to perform load balancing from different access points in one single repository namely, the Network Manager. However, an additional important piece of information is required by the Network Manager to take a suitable decision on which nodes to move, that is, the

visibility of access points to mobile nodes. This requires issuing requests to mobile nodes to retrieve the list of visible access points to each one of them. This cannot be achieved by having an infrastructure topology because additional software will be required on each mobile node to extract the required information and to send it back to the Network Manager. However, based on the partial information gathered by the Network Manager, it is possible to issue a move request to a mobile node, but it could fail, in particular, when the mobile node receives a request to move (from the Network Manager) and the new access point is not visible to it. This limitation is an outcome of the inability of the Network Manager to negotiate a decision with the mobile nodes bound to the loaded access point prior to selecting a candidate node for moving, consequently, it diminishes the practical use of this topology. For example, we could imagine a scenario with many mobile nodes losing their connections due to move requests to access points that are not visible to them, leading to an overall reduction in the network throughput. However, it is possible to enhance the Network Manager with additional logic that infers whether a mobile node was successful in associating with the new access point or not. For example, the Network Manager could store the MAC addresses of the newly moved mobile nodes in a cache until a packet is monitored with the moved mobile node as its source and accordingly, the Network Manager infers that the move was successful. If that is not detected within a certain time frame, the Network Manager concludes that the mobile node was unable to associate with the new access point. Therefore, the Network Manager can undo the changes by restoring the original privileges of the mobile node on its old access point. Clearly, this is a complication to this topology and in fact it will only work properly, if the mobile node sends packets through the gateway that runs the Network Manager. However, the advantage of this approach is its transparency; mobile nodes are not required to run any additional software to interact with the Network Manager, furthermore, there is no additional overhead on the wireless part of the network as all packets are exchanged between the Network Manager and the access points through the backbone.

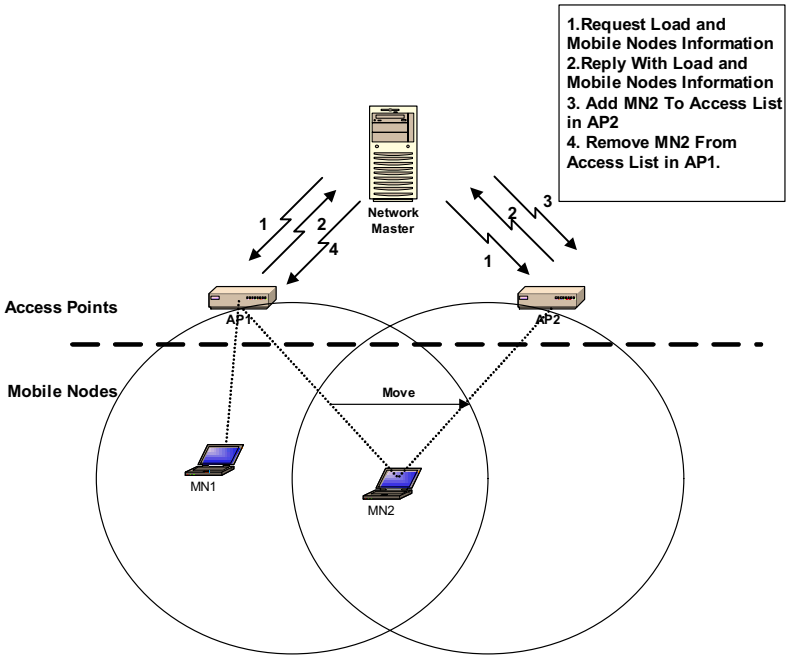

Figure 6: Example Component Interaction, Infrastructure Topology

We now give a detailed description of the control protocol between the Network Manager and the access points it manages.

3.2.1.2 Network Manager Control Protocol

REMOVE_ACCESS_LIST

When the load on an access point increases above a specific threshold, the Network Manager sends a management message (such as SNMP) to the current access point of a selected mobile node to remove its MAC address from the list of nodes allowed to use the access point as a bridge. This will alter the configuration of the access point, therefore requiring it to reinitialize. Reinitializing might take significant time, for example, Orinoco AP-1000 reinitializes itself by reloading the kernel, which takes nearly 10 seconds.

ADD_ACCESS_LIST

When the load on an access point increases above a specific threshold, the network manager sends a management message (such as SNMP) to the new access point (that overlaps with the old one) to add the MAC address of the selected mobile node to its access list, therefore allowing the mobile node to bind to the new access point and to send packets through it. Similar to REMOVE_ACCESS_LIST, this action alters the configuration of the access point, therefore it requires reinitializing the access point.

GET_MOBILE_NODES_LIST (optional)

The Network Manager uses this message to extract a list of mobile nodes connected to an access point. This message is optional as explained earlier since we could simply monitor the network to infer this information.

GET_LOAD (optional)

The Network Manager uses this message to query access points about their load, for example, retrieving the number of incoming or outgoing packets through the access point. This message is optional. It is possible to infer load information by monitoring the network.

3.2.1.3 Example Component Interactions

Figure 6, illustrates the infrastructure topology (with no additional mobile node support) for Sabino. Two access points (AP1 and AP2) are available and are managed by the Network Manager. Stations MN1 and MN2 are initially connected to access point AP1. The manager continuously probes the access points for information about the load in each cell and the list of mobile nodes connected to them (1 and 2). It is important to emphasize that this interaction might not be required, for example, as explained earlier, we could setup the Network Manager as a gateway to which all access points are connected and run on it a monitoring module to infer load information, thus eliminating

the need for probing access points for such information. When the load exceeds a certain threshold, a selection algorithm gets triggered at the Network Manager: it conducts an analysis of the load in each cell using the latest gathered statistics. In addition, it evaluates the possibility of moving one or more mobile nodes to a different overlapping cell in order to improve the overall system performance. Based on the information available to the selection algorithm, a selection is made (in our figure, MN2 is selected). The Network Manager issues a REMOVE_ACCESS_LIST (4) command to the old access point (AP1) of the mobile node (MN2) and at the same time issues an ADD_ACCESS_LIST (3) command to the new access point (AP2). After this action, the packets sourced from or destined to the mobile node (MN2) will be denied access to the old access point (AP1). After a certain period of time, MN2 will detect that the connection has been lost, consequently, MN2 triggers a *sweeping* phase to search for and associate with a new access point as explained in section 2.1. Clearly, the only access point that will allow that association is AP2. As a result, roaming will take place and the load on AP1 will decrease provided that MN2 is in the vicinity of AP2. In case, MN2 is unable to associate with AP2 for any reasons, the Network Manager will undo the actions after a specific time frame, particularly, when it fails to detect any packets sourced from the mobile node that just moved. The Network manager will attempt to select different mobile nodes to move instead of the ones that failed.

3.2.2 No-Infrastructure Topology

The second architectural choice that we explore is based on a distributed architecture that requires no infrastructure support. Each mobile node runs a Mobile Node Controller module that is responsible for monitoring the load in the current cell, negotiating decisions with other mobile nodes and issuing commands to the wireless interface to associate or dissociate with access points. We describe the main component of this topology, namely, the Mobile Node Controller.

3.2.2.1 The Mobile Node Controller (MNC)

The main component in this topology is the MNC: every mobile node runs an instance of this software. It is designed to communicate with access points to extract load information. The MNC keeps track of the load information of the access point that the mobile node is associated with. Unlike the central topology explained earlier, load information of access points is not gathered in any single entity in the system, instead it is distributed across the mobile nodes in the coverage area. In addition, MNCs must interact with each other in order to select a mobile node to move from heavily loaded cells to less loaded ones. This negotiation takes place if the load in any cell exceeds a specific threshold. It is worth noting that we do not require running any additional software on the infrastructure to support that functionality, in fact MNCs only use the primitives supported by the network components such as SNMP. In addition, the MNC includes a module that controls the operation of the wireless interface attached to each mobile node. This module issues association and dissociation requests to the MAC controller during roaming, also, it can extract the list of access points visible to the mobile node and any additional information required such as the signal strength of the beacons arriving from

visible access points. Since that topology does not require any support in the infrastructure, deploying it might be more economic, however, it assumes that it is possible for mobile nodes to retrieve load information from access points which might require some privileged access to the infrastructure, therefore, creating security hazards.

There are several limitations to this topology. The mobile node and the operating system running on it must provide drivers or management APIs that could control the associations and dissociations made by wireless interfaces. Such support is uncommon in current systems, for example, the latest Linux release (kernel 2.4.18) does not support such functionality. However, Windows XP provides specialized APIs for that kind of control [Ayyagari, 01], [Bahl, 00a], [Bahl, 00b]. Another limitation results from the negotiation process that takes place in selecting a mobile node for moving. It creates significant overhead on the wireless part of the network, in particular, mobile nodes that are able to move respond to the negotiation request with packets that include information about the possibility of them roaming to different access points, as a result, these packets add significant overhead on the wireless part of the network. Another critical limitation to this architecture is the inability of mobile nodes to retrieve from the new access points (that they are moving to) load information prior to issuing a handoff, therefore, it is possible that a mobile node switches to a new access point that is already heavily loaded and as a result, reducing the throughput of the new cell and possibly the whole network. The hybrid architecture that is presented later overcomes this limitation by having additional minimal infrastructure support.

We describe the interaction protocol required between different system components to implement a distributed topology.

3.2.2.2 MNC Interaction Protocol

GET_LOAD

The MNC uses this message to query access points about their load, for example, retrieving the number of incoming or outgoing packets through the access point. This message gets issued periodically, in particular, after the mobile node waits for a *backoff interval* that consists of a constant interval of time and an additional random interval. The constant interval controls how many messages could be issued in a certain time frame, while the random interval reduces the possibility of having multiple mobile nodes issuing GET_LOAD requests simultaneously.

LOAD _REPLY

This message returns the load information on an access point at a given time. Typically, access points keep track of the packets going through them in a Management Information Base (MIB), in addition, they export interfaces to allow the retrieval of such information. However, this might require privileged access, for example, in Agere Orinoco AP-1000 access points, it is necessary to have access to an SNMP password to be able to retrieve this information.

BROADCAST_LOAD

The MNC broadcasts a UDP message with the load information that it just received (from a LOAD_REPLY message) to all mobile nodes in the cell. The message is sent to an IP multicast group that contains all the IPs assigned to the mobile nodes. Upon receipt of a BROADCAST_LOAD message, MNCs update their cached load information, and reinitializes their backoff timers that determine when the next load update is to be received.

SWITCH_REPLY

The MNC broadcasts a UDP message to the IP multicast group indicating that it is possible for it to move to an overlapping cell. The message includes the access point to which the mobile node wants to associate. This message is issued after receiving a BROADCAST_LOAD message or a LOAD_REPLY message with the load information exceeding a specific threshold. If the mobile node is unable to move to an access point, it ignores the BROADCAST_LOAD or the LOAD_REPLY messages.

BROADCAST_SWITCH

The MNC that decides to switch notifies all the mobile nodes about its selection using a BROADCAST_SWITCH message. This is done by sending a UDP broadcast packet to an IP multicast group that includes all the mobile nodes. Upon receiving this message, MNCs add extra delay to the backoff interval of the next GET_LOAD message to allow the load on the access point to decrease after the mobile node associates with the new access point. This avoids the situation of having unnecessary mobile nodes switching while the load is decreasing.

3.2.2.3 Example Component Interaction

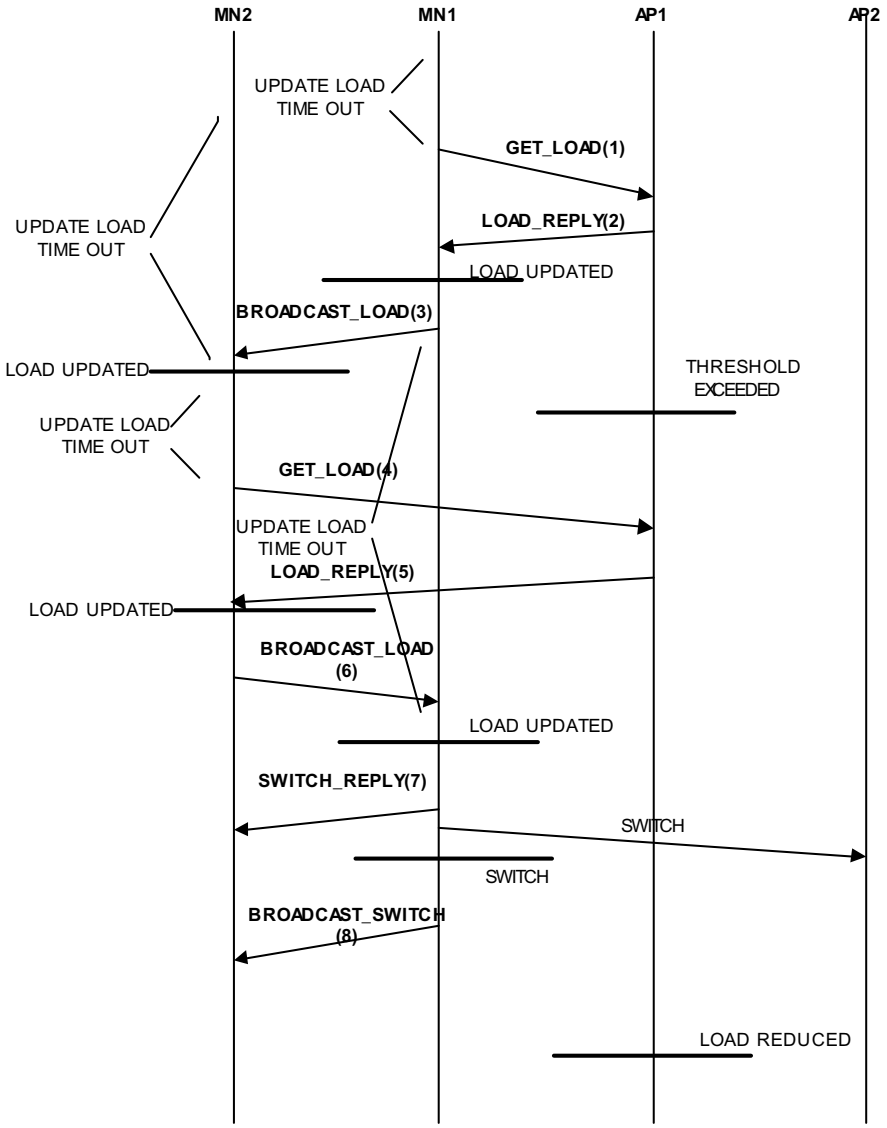

Figure 7: Example Component Interaction, No-Infrastructure Topology

Figure 7 illustrates an interaction in a typical distributed operational scenario with no infrastructure support. The network is configured with two overlapping cells (AP1 and AP2). Initially, two mobile nodes (MN1 and MN2) are bound to access point AP1. Each mobile node maintains a local backoff timer that is initialized to the sum of a constant interval and a random interval, in our interaction scenario we initially assume that the backoff interval of MN1 is less than the interval of MN2, when the backoff interval of MN1 reaches zero it sends a GET_LOAD (1) message to the access point it is attached to (AP1). The access point replies with a LOAD_REPLY (2) message that includes the required load information on the access point, for example, it could indicate the number of outgoing and incoming packets during a specific time frame. MN1 receives the

information and stores it locally, in addition, it issues a BROADCAST_LOAD (3) message to all the mobile nodes in the IP multicast group and finally it reinitializes its backoff timer for the next round of interaction. Similarly, when MN2 receives the BROADCAST_LOAD (3) message, it updates its local cached load information and it reinitializes its backoff timer. In the next round of interactions, the access point (AP1) load increases above a specific threshold, MN2 consumes its backoff interval before MN1, therefore, it issues a GET_LOAD (4) message to the access point (AP1). AP1 replies with a LOAD_REPLY (5) message showing the increase in load. Similar to the first interaction MN2 broadcasts the load information to update the caches of all mobile nodes, however, the MNC will notice the increase in the load, therefore, mobile nodes that can move will reply with a broadcast SWITCH_REPLY (7) message. SWITCH_REPLY messages are preceded with a random backoff element that avoids switching multiple nodes at the same time. It is worth noting that mobile nodes select the access point to switch to using the available signal strength information. The SWITCH_REPLY(7) message informs all mobile nodes that there is indeed a node that is switching and therefore, they need not switch or reply to the BROADCAST_LOAD (6) message. Finally, when a mobile node moves (MN1), it must issue a BROADCAST_SWITCH (8) message to all mobile nodes to indicate that a handoff has taken place, consequently, mobile nodes must add additional delay to their GET_LOAD backoff timers to allow the load on the access point to decrease.

3.2.3 Hybrid Topology

The third architectural choice that we explore is a hybrid of both previous topologies. The hybrid topology requires support on both the mobile node and the access points. It consists of two main components:

I. The Cell Manager. An instance of a cell manager runs on each access point in our system. The component monitors the load within a cell, exchanges this load information with other cell managers and issues control commands to mobile hosts within its cell in order to achieve load balancing within the system.
II. The Mobile Host Control Module (MHCM). An instance of the MHCM runs on each mobile host in our system. The component controls the host's wireless PC card and communicates with the cell manager, supplying the cell manager with information about the host and responding to control commands issued by the cell manager.

Typically cell managers gather load information from MHCMs and other instances of cell managers and based on this information issue commands to MHCMs to disassociate from their current access point and reassociate with a different, less heavily loaded, access point.

It is important to stress that it is not necessary for our components to be installed on all access points and mobile nodes in a given network. Rather, the more widely deployed

our components are, the more effective the system will be at improving overall system performance.

3.2.3.1 The Cell Manager

The cell manager is responsible for monitoring load on its access point and, when this load exceeds a specified threshold, determining which, if any, of the currently registered mobile hosts should be asked to move to a new access point providing overlapping coverage. Our system design is agnostic with respect to the precise method used for determining the current load in a cell. For example, this information is currently available from most access points via management interfaces or it could be obtained by monitoring traffic on the access point's network segment.

In order to achieve load balancing the cell manager also requires knowledge of the load on access points providing overlapping cells and the visibility of these access points to mobile hosts. The knowledge of the load on access points providing overlapping cells is obtained by observing periodic announcements from neighboring cell managers. Our system does not demand that this information be consistent across all access points since the figures provided are used only as an indication of an access point's current load rather than an absolute measure. We describe the protocol we use to maintain this state in section 3.2.3.2.

The final piece of information required by the cell manager is a list of visible access points from each mobile host currently associated with the cell manager's access point. When the load on its access point increases beyond a specified threshold, the cell manager issues a request to the mobile hosts associated with the access point to solicit from each a list of the access points visible to the host together with the signal strength associated with each access point.

Based on the information that the cell manager has gathered from the access points, it is possible to determine the overlapping cells that have significantly less load. Assuming that such an access point exists, the cell manager runs a decision algorithm whose task is to select one or more clients to move to one or more of the overlapping cells (each node can only be associated with a single access point at any point in time). Clearly, the selection process must take into account the signal strength detected by each client for these candidate destination cells, along with the information characterizing the traffic demands of each mobile node. The aim is to reduce the load on the current cell to a level at which it is not significantly impacting on mobile node's communications while avoiding overloading neighboring cells or moving nodes unnecessarily. In our current implementation, we use a simplistic decision algorithm that selects the candidate mobile unit that reports the strongest signal strength for an overlapping cell. We describe the algorithm in more detail in section 4.2.1.

Once the cell manager decides on one or more clients to move, it negotiates this decision with the overlapping cells. The cell manager checks with the destination cell managers to confirm their willingness to host the candidate mobile units. This avoids the

problem of overloading destination cells or moving clients to destination cells that reject the creation of the new association. Finally, the cell manager confirms the movement with the destination access point(s) and issues a move request to the mobile nodes (who may or may not choose to implement the request).

3.2.3.2 The Cell Manager Coordination Protocol

The cell manager coordination protocol is used to exchange information and control messages between cell managers running on different access points. The principle messages supported are:

BROADCAST_LOAD

The cell manager periodically broadcasts UDP messages with the latest monitored load on its access point along with the logical network name of the access point. The message is sent to an IP multicast group that contains all of the cell managers that wish to coordinate their activities. Typically, this would include all cells within a given installation or administrative domain. If the number of cells is very large separate groups can be created since only those cells that might possibly overlap need exchange information. Upon receipt of a BROADCAST_LOAD message, cell managers update their cached load information for the broadcasting access point. Requiring cell managers to periodically broadcast load information maintains the soft state of the whole network load in each cell manager.

SOLICIT_LOAD_INFO

The SOLICIT_LOAD_INFO message is sent by cell managers when they wish to rebuild their cache contents without waiting for periodic messages from all of the other cell managers in the group. The most obvious application of this is when an access point recovers from a failure or reboot. Cell managers receiving a SOLICIT_LOAD_INFO message respond with a BROADCAST_LOAD message. Individual cell managers wait a random period before sending this message to minimize the possibility of collisions.

HOSTING_REQUEST

Prior to issuing a move request to a mobile node, the cell manager must send a request to the destination access point's cell manager. This message includes load information for the mobile nodes to be moved. The destination cell manager should respond to this message with a HOSTING_RESPONSE message.

HOSTING_RESPONSE

Upon receipt of a HOSTING_REQUEST message, the cell manager examines the latest load information captured by its local monitoring module. Assuming that the load in the cell is below a predetermined threshold, and consequently is able to accommodate more mobile nodes, a HOSTING_RESPONSE is sent to the cell manager that issued the HOSTING_REQUEST with an approval flag. If the cell is heavily loaded and cannot accommodate more clients, a HOSTING_RESPONSE is sent to the cell manager that issued the

HOSTING_REQUEST with a denial flag. If the cell manager responds positively, it is assumed that it will reserve resources for the incoming mobile node. More specifically, cell managers should not agree to accept an arbitrary number of incoming mobile nodes but should keep track of the total load they are likely to experience when the nodes complete their move.

HOSTING_CONFIRM

Once a cell manager has received a positive HOSTING_RESPONSE it should confirm the move with the cell manager that has been selected as the final destination. This two phase protocol allows cell managers to enquire about mobile node hosting with multiple possible destinations and then confirm the status of the move with the final destination cell. As an optimization cell managers can send negative confirm messages to any cells that were enquired of but not finally selected to ensure any reserved resources are released in a timely fashion.

3.2.3.3 The MHCM

The MHCM suite is designed to run on mobile hosts in our system. The MHCM interfaces with the PC card installed on every mobile unit, gathering the required statistical information and controlling the association and the dissociation with access points. In more detail, the MHCM issues low-level control commands to query the PC card MIB (Management Information Base) about current signal information and visible access points. In addition, the MHCM issues configuration commands to change the access point to which the client is bound.

3.2.3.4 Cell Manager/MHCM Control Protocol

GET_ACCESS_LIST

When the load on an access points increases above a specified threshold, cell managers send a UDP broadcast message to all mobile nodes in their cell asking them to retrieve the list of access points visible to them.

SEND_ACCESS_LIST

Each mobile node is continuously listening to requests from their cell managers. Upon receipt of a GET_ACCESS_LIST request the mobile node issues a command to the network adaptor and retrieves a list of the visible access points along with their signal strengths. It formats the information and sends it back to its cell manager using a SEND_ACCESS_LIST message.

MOVE_REQUEST

When a mobile node receives a MOVE_REQUEST, it attempts to move to the logical network name associated with the request. If the move fails the mobile host may respond with a MOVE_RESPONSE message.

MOVE_RESPONSE

If a mobile node receives a MOVE_REQUEST message and the move fails the mobile host may respond with a MOVE_RESPONSE message indicating the cause of this failure. Since the cell manager can determine whether or not the mobile node successfully moved independently of this message, this is simply an optimization to provide further information to the cell manager in the case of failure. Note that we do not, of course, send a positive MOVE_RESPONSE message since the mobile node may not be able to communicate with the instigating access point once it has completed the move operation.

3.2.3.5 Example Component Interactions

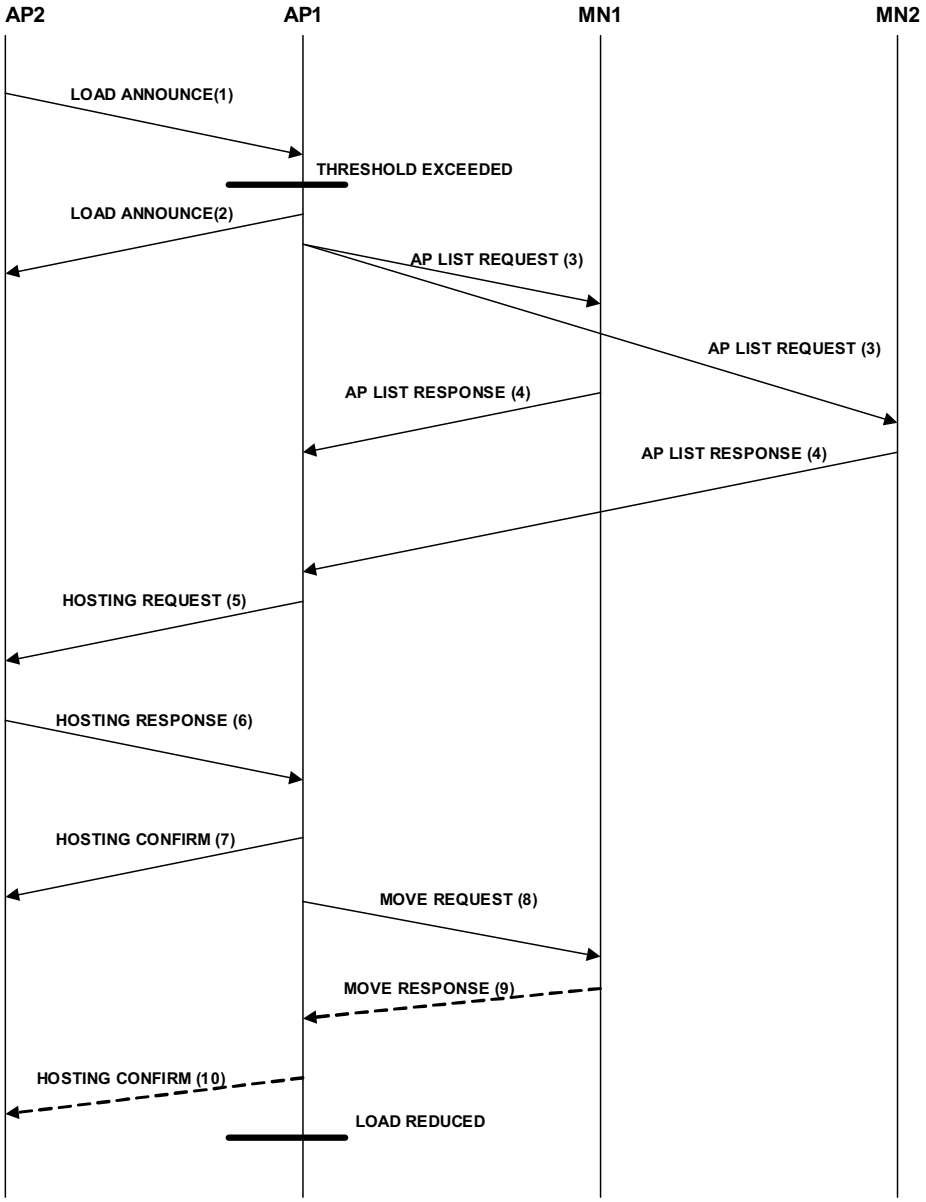

Figure 8: Example Component Interaction, Hybrid Topology

Figure 8 illustrates the interactions between our components in a typical operational scenario. Two access points are shown (AP1 and AP2) along with two mobile nodes (MN1 and MN2), both of which are assumed to be associated with AP1. The cell managers running on AP1 and AP2 periodically broadcast their load information (1). At some point the load on AP1 increases beyond its threshold and it begins the process of attempting to reduce the load below this threshold. Its first action is to broadcast an update of its load information (2) to warn other cell managers that it is not a good target for accepting new mobile hosts. AP1 must then gather information from its mobile hosts regarding the other APs that they can see, and hence could associate with. This information is solicited using a broadcast message to the hosts currently associated with AP1 (3). Mobile hosts respond with a unicast message to AP1 (4).

At this point AP1 has all the information it requires to determine which mobile hosts are candidates for moving to a new cell. Once it has selected a candidate (in this case MN1), it first checks with the target destination cell (AP2) to ensure that it is willing to accept the new mobile unit (5). The destination access point responds with an accept or a reject message (in this case an accept) (6). AP1 can now send a confirmation message to AP2 telling it that the mobile unit will be asked to move (7) and sends a message to the mobile unit itself telling it to move, and specifying the destination AP (8).

Under normal circumstances the mobile node will move and the process will complete. However, the final decision to move rests with the mobile node itself. If it decides not to move it can inform AP1 (9) and AP1 can pass this information on to AP2 (10) allowing AP2 to release any resources it may have reserved for the incoming mobile host. Messages (9) and (10) are optimizations since AP1 will, in any case, be able to observe whether the mobile node leaves its area of coverage. However, a negative response from a mobile host can be used to accelerate the process of releasing resources at the destination access point (AP2) and informing the current access point why the request was rejected.

3.2.4 Discussion

This chapter discussed three different architectural topologies for Sabino, namely, an infrastructure topology with Sabino running on the infrastructure with no additional support from the mobile nodes, a no-infrastructure topology with Sabino running on the mobile nodes with no infrastructure support and a hybrid topology that requires both infrastructure support and support from the mobile nodes. The infrastructure approach has an attractive characteristic, that is, the transparency of the system to the mobile nodes in the coverage area. Association and dissociations are done without the involvement of any of the mobile nodes. One important benefit of this approach is that it does not require the mobile nodes to run any additional software, for example, in practical public access installations, it might not be feasible to require mobile nodes to install Sabino. Therefore, an infrastructure-based topology becomes more suitable as a solution in such domains. However, mobile nodes might suffer from periods of disconnection, in particular, mobile nodes can be forced to switch to access points that are not visible to them. This limitation

is an outcome of the inability of mobile nodes to send the list of access points (visible to them) to the Network Manager. Another limitation to this approach is that it requires every access point to modify its configuration on every handoff. Typically, this requires reinitializing the access point, which might take significant time. The second topological architecture that was discussed required no infrastructure support. This resulted in several limitations, for example, mobile nodes are unable to retrieve load information from access points that they are not connected to, as a result, mobile nodes might take the decision to switch to heavily loaded cells, therefore, reducing the overall throughput of the network. Finally, we introduced a hybrid topology that requires both infrastructure support and mobile node support. The topology overcomes many of the limitations of both preceding approaches, for example, mobile nodes can determine the load information on any access point in the system, in addition, mobile nodes can exchange the list of visible access points with any other node, therefore, it is guaranteed that mobile nodes will switch to access points that are visible to them, as a result, it is no longer possible to have disconnection periods during roaming. It is clear that having a topology that requires both infrastructure and mobile nodes support is advantageous compared to other solutions.

The next chapter discusses two implementations of Sabino: the no-infrastructure topology and the hybrid topology. In addition, the chapter describes the experimental setups and the different components in the implementation.

Chapter 4

Implementation

This chapter describes the implementation of the Sabino system. We have implemented two different architectural topologies: the no-infrastructure topology and the hybrid topology. The two core components in the former topology are the MNC and the mobile nodes interaction protocol. The three core components of the latter topology are the cell manager, the MHCM and the cell Manager/MHCM Protocol. The implementation of the inter-cell manager protocol is a topic for future work.

4.1 Experimental Configuration

Our basic implementation environment is shown in figure 9.

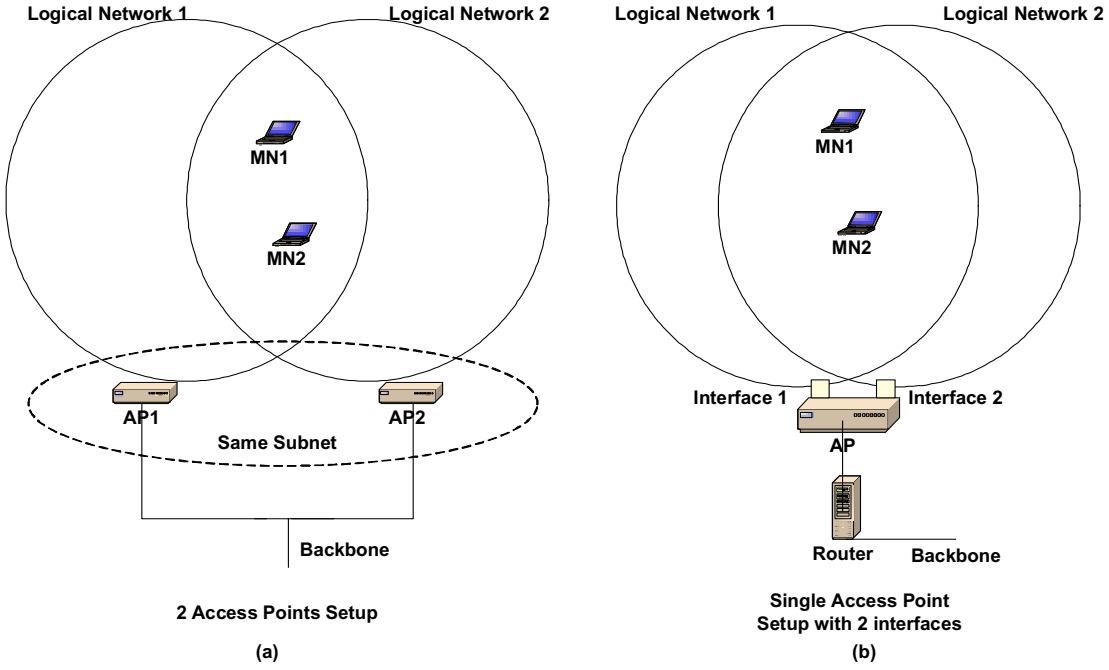

Figure 9: Experimental Setups

In setup (a), we have two Agere Orinoco AP-1000 access points that are configured to provide two overlapping areas of coverage, each utilizing a non-interfering frequency to maximize the throughput. Each access point is given a different logical network name (SSID), this is required to be able to force handoffs, since current cards do not support associations with access points but with logical network names. In setup (b), a single Pentium II 397 MHz machine runs Linux kernel 2.4.2 and acts as a router with two interfaces, one connected to the wired Ethernet backbone and the other connected to an

Agere Orinoco AP-1000. The AP-1000 is configured to provide two overlapping areas of coverage, each utilizing a non-interfering frequency to maximize the throughput. It is configured to work as a bridge with the two wireless interfaces given different logical network names (SSIDs). In both experimental setups, the two logical networks are connected to the same physical subnet, therefore mobile nodes can maintain their IP addresses as they move between logical networks. As a result, we do not require mobile IP in order to maintain connections as mobile nodes move between access points. In an event of a handoff the learning bridges in the access points will adjust their entries to correctly route the packets to their destination. Two IBM Thinkpad laptops, equipped with Agere's Orinoco silver cards, are used in our experiments. One laptop is a Pentium III 697 MHz running Windows XP Professional while the other is a Pentium II 996 MHz running Windows 2000.

Setup (a) is used for the no-infrastructure based topology experiments. Setup (b) is used for the hybrid topology experiments, the router is acting as a host for our cell manager software (that would ideally be placed in the access point) and ensures that the cell manager is capable of monitoring the packets sent by the mobile units and destined to the wired backbone, hence enabling us to gather the necessary load information for the access point.

4.2 System Components

4.2.1 Cell Manager

The cell manager is part of our hybrid architecture, it runs on the router shown in figure 9 (b) and is responsible for collecting information on the network load and visibility of access points to mobile nodes and issuing control commands to mobile nodes. The cell manager consists of two separate threads that run in parallel. The first is responsible for capturing network load information and utilizes the libpcap library. This thread sets the Ethernet interface connected to the access point in promiscuous mode enabling it to monitor all the packets coming through the interface. Based on this information, the monitor thread updates a cache that keeps track of the load information of each mobile unit bound to the access point. Each cache entry includes the relative load of each mobile unit and its IP address. An alternative implementation approach would be to obtain the load information from the access point itself similar to the approach presented later in the no-infrastructure topology, particularly, using SNMP management packets.

The cell manager's cache is flushed and refreshed every 1 second to keep the information up-to-date. Prior to flushing the cache, the monitor thread initiates a performance evaluation check that uses the information populating the cache to ensure that the overall load on the access point is below a certain threshold. If the load exceeds the threshold it begins the process of selecting an appropriate mobile node and issues commands to the mobile node to move to a new access point. In our current implementation, we are using a simple approach for selecting the candidate mobile node based on the signal strength detected at each node. More specifically, we simply move

the client that reports the best quality signal to an alternative access point that is not itself excessively loaded. Clearly this algorithm could be the subject of much investigation. For example, it is unclear how to trade-off access point visibility against load when considering whether to move a mobile node.

The second thread in the cell manager is the mobile listener thread. This thread captures the information sent by the mobile units during the negotiation process and populates the cell manager's cache with this information.

4.2.2 The MHCM

The MHCM is the second main component in the hybrid topology. It runs on mobile units. The MHCM relies on features from the underlying operating system and network adaptor to provide information such as the list of currently visible access points. Currently these features are only supported in Microsoft Windows XP [Ayyagari, 01] [Bahl, 00a] [Bahl, 00b]. In particular, a number of new OIDs (Object Identifiers) are available via Windows Management Instrumentation (WMI) and are required to be supported by Network Interface Card (NIC) drivers in XP environments while being optional for Windows 2000 drivers. These OIDs provide the ability to retrieve the list of all access points detected (including attributes such as the signal strengths and the MAC addresses), to cause the NIC to request a survey of potential access points using active or passive scanning, to associate, dissociate or re-associate the network adaptor with a different logical network and to associate, dissociate or re-associate the network adaptor with a specific access point (not yet supported).

The MHCM runs as a single thread of execution that listens to a specific port for cell manager requests. When it receives a message it performs the action required and replies to the cell manager. In our current prototype, we have implemented the following operations:

I. retrieving the list of access points visible to the mobile host along with their signal strengths, and,
II. switching to a new logical network specified by a logical network name.

It is worth emphasizing at this point that the MHCM can only switch between logical networks and not between access points within the same network. This explains the configuration shown in figure 9 (b) in which the access point's two network interfaces are configured with separate network names. One practical implication of this is that clients are unable to roam between the networks since 802.11 only supports roaming within a single logical network. However, our set-up enables us to evaluate the feasibility of our ideas until such point as support for the association with a specific access point is provided by the firmware.

4.2.3 The MNC

The MNC is the main component in the no-infrastructure topology, it is similar to the MHCM in the hybrid topology but with some modifications and some additional features. Similar to the MHCM, the MNC relies on features from the underlying operating system and network adapter, in particular, the APIs provided by Microsoft Windows XP that can control the wireless interface as explained earlier. The MNC runs with two threads of execution: the first listens to a specific port for mobile node broadcasts, it is responsible for updating the cached load information of the access point and responding for move requests if the load exceeds a specific threshold. The second MNC thread is responsible for issuing SNMP requests to the access point that the mobile node is connected to. The MNC uses winsnmp2 libraries in Windows XP to construct messages with OID codes that query the number of packets that have passed through the access point interfaces. When an access point receives an SNMP message it retrieves the count of the incoming and outgoing packets from the MIB, and sends the information back to the requesting node. After retrieving the information, the SNMP thread broadcasts the load to all mobile nodes serviced by the access point. It is worth noting that AP-1000 access points maintain a single packet count entry for both of its wireless interfaces, therefore, it is not possible to isolate the load on each interface using SNMP. This explains the configuration shown in figure 9 (a) in which two access points are required to be able to obtain load information in two separate cells using SNMP messages.

The next chapter discusses the experiments that were conducted to evaluate Sabino as an architectural solution.

Chapter 5

Analysis

We have conducted a series of initial experiments with our prototype system to validate the Sabino system.

5.1 Load Balancing in Existing Systems

As mentioned earlier, the Agere Orinoco system supports an implicit load balancing scheme. If a mobile node misses multiple consecutive beacons from its current access point, it attempts to locate a new access point to associate with [Lucent, 98a]. The loss of beacons could be caused by any number of factors such as interference or access point failure or, according to, the network becoming heavily loaded. To test whether this feature operates correctly, we performed two simple tests in which we carried out two parallel 40 MB FTP transfers to two mobile nodes connected to the same access point using the setting shown in figure 9 (b) and we carried out another two 60 MB FTP transfer to two mobile nodes connected to the same access point using the setting shown in figure 9 (a). An overlapping cell with the same logical network name was available for the nodes to associate with if they desired in both settings. If the existing load balancing system was working correctly we would expect that one of the mobiles would switch to this second access point as the initial access point becomes heavily loaded.

In practice, we found that despite extremely heavy load in the initial network, at no point did either of the nodes switch to the alternative, and otherwise unloaded, access point. Table1 and Table2 provide figures for the typical transfer rates we obtained in these scenarios. We conclude therefore that current load balancing schemes are not effective in practical network settings.

Node	File Size	Transfer Time (sec)	Data Rate (Kbps)
MN1	40 MB	132.21	2440
MN2	40 MB	133.2	2422

Table 1 : File transfer measurements using the standard load balancing scheme with 1 access point

Node	File Size	Transfer Time (sec)	Data Rate (Kbps)
MN1	60 MB	261.17	1840
MN2	60 MB	258.47	1864

Table 2 : File transfer measurements using the standard load balancing scheme with 2 access points

5.2 Roaming Overheads

Clearly, the success of any load balancing system is heavily dependent on the time taken for a mobile node to disassociate from one access point and reassociate with a new access point, i.e. the handoff overheads. To measure this overhead, we created a test environment with two access points each with distinct logical network names (SSIDs). We created a simple process that streamed UDP packets to clients associated with one of these access points. We then set up a mobile node such that it was initially connected to the other access point and measured the time interval between issuing an OID_802_11_SSID command to the mobile host's NIC telling it to switch networks, and the point at which we received the first packet from the streaming server. After repeated tests, we found that this interval was approximately 1 second. It should be noted that the mobile node was able to maintain its IP address when it moved to the new network and hence this figure does not include any overheads that would be incurred in a system that used mobile IP. While a handoff time of 1 second is not negligible, we believe that in most practical applications the benefits of moving to a less heavily loaded cell will quickly outweigh this cost. In the following section we provide evidence to substantiate this claim.

5.3 Performance Gains From Using the Sabino System

In order to test whether significant benefits could be accrued from load balancing in a practical network setting, we re-ran the tests described in section 5.1 *Load Balancing in Existing Systems* but this time with our software installed on the mobile nodes. We also set the two overlapping cells to have distinct logical network names in order that we could effect a move between them. With the existing systems we observed that it took approximately 133 seconds for each mobile node to download a 40 MB file in the setting shown in figure 9 (b). With the Sabino system operational the cell manager quickly detected the overloaded state of the network and reacted by requesting that one of the mobile nodes move to the overlapping cell. This resulted in an average download time of 81 seconds for each mobile node. Table 3 summarizes these results.

Node	File Size	Transfer Time (sec)	Data Rate (Kbps)
MN1	40 MB	81.63	3952
MN2	40 MB	81	3978

Table 3 : File transfer measurements when using the Sabino system with 1 access point

In the setting shown in figure 9 (a), we observed that it took approximately 260 seconds for each mobile node to download a 60 MB file. With Sabino running, the mobile nodes quickly detected the overloaded state of the network and reacted by negotiating the move of one node to the overlapping cell. This resulted in an average download time of 130 seconds for each mobile node. Table 4 summarizes these results. In this experiment, MN2 roamed from an access point that serviced no nodes except MN1 and MN2 to an access point that is used by many other users, this explains the difference in transmission times between MN1 and MN2.

Node	File Size	Transfer Time (sec)	Data Rate (Kbps)
MN1	60 MB	116.63	4120
MN2	60 MB	143.93	3336

Table 4 : File transfer measurements when using the Sabino system with 2 access points

5.4 Overhead of Rescanning to Obtain Access Point Visibility Information

The final issue with which we were concerned is the effect of issuing rescanning requests to the NIC adapters. We created a simple experiment to measure this by transferring a 40 MB file to a server without issuing any rescan requests. This measurement provided a baseline for subsequent measurements. We then re-ran the same experiment while periodically invoking OID_802_11_BSSID_LIST_SCAN operations. We varied the interval between issuing consecutive scanning requests and measured the file transfer time. We observed a significant effect associated with issuing scans: our measurements show a 15 % reduction in the data rate when we issue scan requests every 1 second. Table 5 shows the effect of invoking rescan operations with different intervals. Even when the scan requests are issued every 5 seconds, as suggested by Microsoft MSDN documentation's description of the call, the data rate is reduced by approximately 6%. This reduction can be attributed to the fact that the Orinoco cards we are using implement only active scanning, as described in section 2.1. We have yet to explore the effect of multiple hosts issuing active scans within a similar geographic area.

Scan Interval (sec)	Data Rate (Kbps)
1	3948
2	4196
3	4218
5	4250
9	4297
13	4406
no scan	4622

Table 5 : Impact of scanning for access points on data transfer rate

It is clear that Sabino was able to effectively perform load balancing across several overlapping access points. Sabino outperforms the mechanisms used in commercial products. In the next chapter we explore the potential for future work.

Chapter 6

Conclusion

6.1 Discussion and Future Work

One of the advantages of Sabino is that it can be incrementally deployed. More specifically, it is not necessary for every access point or every mobile node in the system to have Sabino software installed. A minimum installation would be two access points running our software and one mobile node. As the number of access points and mobile nodes that support our software increases, so do the potential performance gains. This clearly has important benefits in terms of the viability of deploying the Sabino system in any real-world environment. The only configuration that must be carried out is the creation of a multicast group that contains all of the cell managers in a system or the mobile nodes serviced by access points. The Sabino system is also largely independent of any decisions taken regarding mobility support at the network level. Specifically, in our test environment we did not run mobile IP but the system would have performed in the same fashion if mobile nodes were to run mobile IP. The only important factor that would need to be taken into account is that the overhead of switching to a new cell may increase and this would need to be reflected in the cell manager and the mobile nodes decision-making algorithms.

It is important to emphasize that load balancing is one of many other attributes that Sabino can optimize. For example, 802.11 supports two classes of medium access control as described in chapter 2. The 802.11 standard suggests alternating between these two modes of operation, namely DCF and PCF. Sabino could effectively be used to dynamically coordinate the switching of these two medium access schemes according to the information gathered by the cell managers. Another potential area for using Sabino is the integration of the system with the QoS primitives supported in 802.11 networks. IEEE 802.11e introduces a set of priority classes that are part of QoS enhancements in the standard. These schemes are designed to support QoS at the cell level, yet fail to provide a global view of the whole network. Integrating Sabino in such environments would provide another dimension for managing the network. As a specific example, different overlapping cells could be established, each tailored for specific classes of traffic. Sabino could then be used to switch mobile nodes to the most appropriate cell based on their traffic profile. In general, we envision Sabino as a lightweight layer that provides mechanisms for managing network resources at an inter-cell level. Thus one important challenge is to combine this inter-cell management with existing and emerging intra-cell schemes.

Furthermore, it is important to evaluate the scalability of Sabino in practical environments, for example, Sabino can be installed in public access networks or across several overlapping networks that use different MAC protocols. In addition, Sabino must

be subjected to a variety of traffic such as streaming applications, ftp transmissions and telnet sessions, as a result, we will be able to better evaluate the implementation. Also, the way by which mobile nodes are selected for switching is subject of further investigation. For example, it is unclear how to trade-off different parameters such as the visibility of access points or the load contributed by each mobile node when considering whether to move a mobile node or not.

6.2 Concluding Remarks

The Sabino system demonstrated that existing 802.11 systems fail to make optimal use of the available network resources because of a lack of coordination between the various system components. In particular, the absence of any form of coordination between access points means that wireless network installations may not make the optimal allocation of mobile nodes to access points, leading to imbalances in the load distribution and potentially reducing the network bandwidth available to users. The solution presented in this thesis takes the form of a lightweight management architecture that coordinates the actions of access points and mobile nodes to ensure, where possible, a more even load distribution and hence better utilization of system resources. Key aspects of the Sabino architecture include easy and incremental deployment, low overhead when system management is not required and effective management of resources when the network becomes loaded. The results presented have shown that Sabino offers significant benefits over existing load balancing systems when deployed in a simple experimental setting. Possible future work is to validate the architecture within the context of a larger network installation and to explore the relationship between inter and intra cell management.

References

[Ayyagari, 01] A. Ayyagari and T. Fout. “Making IEEE 802.11 Networks Enterprise-Ready”. White Paper. Microsoft Corporation. May, 2001.

[Bahl, 00] P. Bahl, A. Balachandran and S. Venkatachary, “The CHOICE Network — Broadband Wireless Internet Access in Public Places,” Microsoft Research Tech. Rep. MSR-TR-2000-21, February 2000.

[Bahl, 00a] Bahl, Paramvir. “Enhancing the Windows Network Device Interface Specification for Wireless Networking”. White Paper. Microsoft Research. 2000.

[Bahl, 00b] Bahl, Paramvir and G. Holland, “Enhancing the Windows Network Device Interface for Wireless Networking Part II”. MSR Technical Report MSR-TR-2000-84. August 2000.

[Caceres, 96] R. Caceres and V. Padmanabhan. “Fast and Scalable Handoffs for Wireless Internetworks”, *Proc. ACM MobiCom '96*, November 1996.

[Comer, 95] Comer, D., J. Lin, and V. Russo. "*An Architecture For A Campus-Scale Wireless Mobile Internet*", Technical Report CSD-TR 95-058, Purdue University, Computer Science Building, West Lafayette, IN 47903-1398. 1995.

[Deng, 99] J. Deng and R-S. Chang. “A Priority Scheme for IEEE 802.11 DCF Access Method”. IEEE Transactions on Communications. Vol.E28-B. No 11. January 1999.

[Friday, 01] A. Friday, M. Wu, S. Schmid, J. Finney, K. Cheverst and N. Davies, “A Wireless Public Access Infrastructure for Supporting Mobile Context-Aware IPv6 Applications,” In *Proc of WMI 2001*, Rome, Italy. 2001.

[Godfrey, 01] Godfrey, Tim. “802.11 Task Group E Status”. Intersil Coorporation. Presentation to the OFDM forum San Francisco. February, 2001.

[Hills, 96] Hills, A. and D. B. Johnson, "A Wireless Data Network Infrastructure at Carnegie Mellon University," *IEEE Personal Communications,* vol. 3, no. 1, pp. 56-63, February 1996.

[IEEE, 99a] IEEE. “IEEE Standard for Information Technology: Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”. 1999 Edition.

[IEEE, 99b] IEEE. “Supplement to IEEE standard for Information Technology: Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications”. September, 1999.

[IEEE, 01a] IEEE. “IEEE P802.11 Wireless LANs: IEEE 802.11f pre-Draft”. IEEE. January, 2001.

[IEEE, 01b] IEEE. “IEEE P802.11 Wireless LANs: IEEE 802.11e Draft 2”. IEEE. 2001.

[Katz, 96] R. Katz and E. Brewer. “The case for wireless overlay networks”. In Proc. of SPIE Multimedia and Networking Conference, January 1996.

[Kopsel, 01] A. Kopsel and A. Wolisz. “Voice Transmission in an IEEE 802.11 WLNA based access network”. Proc. of ACM Workshop on Wireless Mobile Multimedia, July 2001.

[Lucent, 98a] “Roaming With WaveLAN/IEEE 802.11” Lucent Technologies Inc. WaveLAN Technical Bulletin 021/A, December 1998.

[Lucent, 98b] “IEEE 802.11 Channel Selection Guidelines” Lucent Technologies Inc. WaveLAN Technical Bulletin 003/A, November 1998.

[McCloghrie, 91] McCloghrie, K. and Rose, M. “Management Information Base for Network Management of TCP/IP-based internets:MIB-II”. March, 1991. RFC, 1213

[Nakajima, 01] Nakajima N. and Yamao Y. "Development of 4th Generation Mobile Communications", Wireless Communications and Mobile Computing 2001, issue 1, pp.3-12. 2001.

[Pradhan, 98] Prashant Pradhan, Tzi-cker Chiueh, "*Real-Time Performance Guarantees over Wired/Wireless LANs*," IEEE Conference on Real-Time Applications and Systems, June 1998.

[Stemm, 97] M. Stemm and R. Katz. "Vertical handoffs in wireless overlay networks. ACM MONET Special Issue on Mobile Networking in the Internet. 1997.

[Wang, 99] H. J. Wang, R. H. Katz, and J. Giese. "Policy-Enabled Handoffs Across Heterogeneous Wireless Networks". In WMCSA 99, New Orleans, LA

[Weinmiller, 96] J. Weinmiller, H. Woesner, and A. Wolisz**. "**Analyzing and Improving the IEEE 802.11-MAC Protocol for Wireless LANs**".** 4th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'96).