Seamless Connectivity and Mobility in Wireless Mesh Networks

by

Nilo Rivera

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

September, 2008

Abstract

Wireless mesh networks extend the connectivity range of mobile devices by using multiple access points, some of them connected to the Internet, to create a mesh topology and forward packets over multiple wireless hops. Mobile clients should be able to freely roam within the area covered by the mesh and maintain their connectivity at all times.

This thesis presents the architecture and protocols of the first transparent wireless mesh system that offers seamless, fast handoff, supporting VoIP and other real-time application traffic for any unmodified 802.11 device. The entire mesh network is seen by the mobile clients as a single, omnipresent access point. Access points continuously monitor the connectivity quality of any client in their range and efficiently share information with other access points in the vicinity of that client to coordinate and decide which of them should serve the client. We first show an intra-domain handoff protocol that transfers connectivity between the access points serving the mobile device. We then show an inter-domain handoff protocol that transfer connectivity between access points connected to the Internet. Both handoffs, which can occur simultaneously, maintain all previously opened connections while transferring them as fast as possible without any involvement from the mobile device. Experimental results on a fully deployed mesh network demonstrate the effectiveness of the architecture and its handoff protocols.

Advisor: Dr. Yair Amir
Readers: Dr. Yair Amir, Dr. Claudiu Danilov, Dr. Andreas Terzis

Acknowledgements

I am deeply indebted to my advisor Dr. Yair Amir whose guidance and support helped me through every stage in my studies. Yair always had very high expectations of my work, and this helped me raise my own and become a better scientist. His personal and professional advice over the years will contribute greatly to my career and my life in general.

I am deeply grateful to Dr. Claudiu Danilov for all his help and advice, both professionally and as a friend, during my Ph.D. His input helped shape my research from the very beginning to the very end. Also, I want to thank Dr. Andreas Terzis for his time when exchanging ideas at the beginning of my studies and for being a reader for my dissertation.

I thank Dr. Russell H. Taylor for giving me the opportunity to work with him on an inter-disciplinary project with the school of medicine. I also want to thank Dr. Randal Burns for his input during and after my oral examination, and for his time while brainstorming on possible research topics. Also, I would like to thank Russ and Randal, as well as Dr. Brinton Cooper and Dr. Stuart Leslie, for being a part of my oral examination committee.

I thank Raluca Musăloiu-Elefteri for helping me generate great ideas and for her support in building a successful system. Also, I want to thank Jonathan Kirsch and John Lane for their input and for taking the time to share their knowledge on areas that interest me as well.

I would also like to thank Dr. Ramesh Bharadwaj for the opportunity to work at the Naval Research Laboratory during the first summer of my Ph.D.. I gained extremely valu-

able knowledge and experience during this time.

I have been fortunate to have interacted with people like Wyatt Chaffee, Jacob Green, Dr. John Linwood Griffin, Michael Hilsdale, Michael Kaplan, Sandeep Ranade, John Schultz, Swaroop Sridhar, and Dr. Ciprian Tutu. Also, the memory of an old friend in life, Raul Sanchez, has given me strength many times in life.

I would also like to thank Dr. Fazil T. Najafi who gave me advice during my years as an undergraduate student at University of Florida and also supported me when applying to graduate school.

I am very grateful to the National GEM Consortium and the Johns Hopkins Whiting School of Engineering for the fellowship that allowed me to embark in my Ph.D studies.

Nobody has been more important to me in the pursuit of this project than the members of my family. I would like to thank my parents, Nilo and Idahlia, whose love and guidance are with me in whatever I pursue. Most importantly, I wish to thank my loving and supportive wife, Claudia, and my children, Nilo Eduardo and Veronica Giselle, who remind me everyday of the beautiful details that life has to offer. I want to thank my sister, Michelle, who is a very special part of my life. Also my mother and father-in-law, Eduardo and Teresa, for their love and support, as well as my sister-in-law, Paola. To my grandparents, always there with love, and to my godfather, Dr. Jose G. Quiñonez, who supported me in every step since my childhood. And last but definitely not least, I would like to thank God for giving me wisdom and guidance throughout my life.

Contents

List of Tables

List of Figures

To my beloved and beautiful wife, Claudia, infinitely supportive.

To my amazing children, Nilo Eduardo and Veronica Giselle, precious miracles of life.

To my parents, Nilo and Idahlia, supportive in every endeavor in my life.

And to my sister, Michelle, always reassuring.

Chapter 1

Introduction

Wireless networks have changed the way people connect to the Internet, giving users the freedom to connect from anywhere within the coverage area of a wireless access point. Wireless Mesh Networks extend the wireless coverage area of an access point by having only a few of the access points connected to a wired network, and allowing the others to forward packets over multiple wireless hops. A mesh networks can span a large geographical area and Internet connected access points (*Internet gateways*) may reside at different network domains, effectively creating a *multi-homed* wireless mesh network.

When a user moves outside the range of an access point and closer to another, it switches its connectivity to the closest access point. This connectivity change involves a transition (*handoff*) before being able to route packets to and from the new access point. Maintaining connectivity requires a handoff at two levels. An intra-domain handoff is required to transfer connectivity between the access points serving the mobile device. At a higher level, an inter-domain handoff between access points connected to the Internet may be required on existing Internet connections. Both handoffs, which can occur simultaneously, must maintain all previously opened connections while transferring them as fast as

possible. Ideally, the handoff should be completely transparent to mobile clients. There should be no interruption in network connectivity, and the communication protocols involved should follow the standards deployed in regular wireless devices. We call a wireless network that offers such a service a *seamless* wireless mesh network.

While cell phone networks solve the handoff problem using signaling embedded in their low-level protocols, there are currently no efficient, transparent handoff solutions for wireless 802.11 networks. Most wireless mesh networks today require specially modified clients in order to transfer connectivity from one access point to the next. Others, even if they give the appearance of continuous connectivity to a roaming client, provide connections that are in fact interrupted when a client transfers from one access point to the next, with delays that can be as long as several seconds. For some applications (e.g. transferring files), this delay is acceptable; however, it is far too long for real-time traffic such as interactive Voice over IP or video conferencing.

This thesis presents the architecture and protocols of the first transparent wireless mesh network that offers seamless fast handoff, supporting VoIP and other real-time application traffic. All the handoff and routing logic is done solely by the access points, and therefore connectivity is attainable by any 802.11 mobile device, regardless of its vendor or architecture. In order to provide this level of transparency to mobile clients, our approach uses only standard network protocols. The entire mesh network is seen by the mobile clients as a single, omnipresent access point, giving the mobile clients the illusion that they are stationary.

A software system called SMesh [1] was created to enable us to pursue the research presented in this thesis with a practical approach. The system was deployed throughout various building at The Johns Hopkins University main campus and made available as

open-source for others to deploy. Our experiments were conducted with real clients moving throughout the SMesh deployment, demonstrating the performance of our protocols in a realistic environment.

1.1 Highlights and Contribution

We present a new architecture and algorithms for providing seamless connectivity and fast handoff to mobile clients. The approach requires that we provide intra-domain handoff when the client moves between access points, and inter-domain handoff when the client moves between mesh nodes connected at different network domains.

Fast intra-domain handoff is achieved by controlling the handoff from the mesh infrastructure and by using multicast to send data through multiple paths to the mobile client during handoff. Mobile clients are handled by a single access point during stable connectivity times. During the handoff transitions, our protocols use more than one access point to handle the moving client. Access points continuously monitor the connectivity quality of any client in their vicinity and efficiently share this information with other access points in the vicinity of that client to coordinate which of them should serve the client. If multiple access points believe they have the best connectivity to a mobile client, and until they synchronize on which should be the one to handle that client, data packets from the Internet gateway (or another source within the mesh network) to the client are duplicated by the system in the client's vicinity.

Fast inter-domain handoff is achieved by using multicast groups through the wired network to coordinate decisions and seamlessly transfer connections between Internet gateways as mobile clients move between access points. New connections always use the

closest Internet gateway at the time of their creation, while existing connections are forwarded through the wired infrastructure to the Internet gateway where they were originally initiated. As the handoff process requires routing agreement and transferring connections between the involved Internet gateways, our protocol guarantees that packets are routed correctly, at all times.

While duplicating packets and tightly coordinating access points in a client's vicinity may seem to incur high overhead, this thesis will quantify the overhead and demonstrate it is negligible compared to data traffic.

We also show how our system supports peer-to-peer communication between mobile clients by providing automatic routing for clients connected to the mesh. The forwarding and coordination between the access points is done using our Spines messaging system [2] that provides efficient unicast, anycast, and multicast communication.

The main contributions of this thesis are:

- The first *seamless* 802.11 wireless mesh network with fast handoff that supports real-time applications such as interactive VoIP and video conferencing.
- A simple and practical architecture that seamlessly integrates wired and wireless connectivity in multi-homed wireless mesh networks.
- Novel use of multicast for localized access point coordination in tracking mobile clients, for robust mesh to client communication during intra-domain handoff, and for communication between Internet gateways during inter-domain handoff.
- Novel use of anycast for mobile client to mesh Internet gateway communication.
- Innovative use of the DHCP and ARP protocols for monitoring connectivity quality of mobile clients and for creating a single, virtual access point throughout the

wireless mesh.

1.1.1 Thesis Organization

The rest of the thesis is organized as follows: The next section overviews related work in wireless mesh networks, intra-domain handoff, and inter-domain handoff. Chapter 2 describes our wireless mesh system, SMesh, and presents its architecture, seamless connectivity and monitoring of mobile clients, and how SMesh transparently routes packets through an overlay network with a generic interceptor. Chapter 3 presents our fast intra-domain handoff protocol, which includes client monitoring, mobility management, and fast handoff approach. In Chapter 4, we present our fast inter-domain handoff for multi-homed wireless mesh networks and how TCP and UDP connections are separately handled to correctly route these packets. Chapter 5 summarizes our contribution and concludes the thesis.

1.2 Related Work

Much of the work on handoffs in 802.11 wireless networks is essentially trying to duplicate the successful handoffs that already exist in cell phone networks when a mobile device roams between towers. By requirement, a cell phone handoff must be quick enough to support full-duplex voice communication without a perceivable gap in either voice stream.

Seamless mobility in wireless mesh networks must account for movement at two different levels: intra-domain, between access points, and inter-domain, between Internet connected access points potentially connected on different networks. As such, our work relates to previous work on wireless mesh networks, intra-domain handoff, and inter-domain handoff. In addition, our approach benefits from the rich set of services overlay networks

provide.

Good surveys addressing all of these areas were overviewed by Akyildiz et al. in [3] and [4]. Note that related work may also refer to intra-domain handoff as *micromobility* and to inter-domain handoff as a form of *macromobility*.

1.2.1 Wireless Mesh Networks

There has been a considerable amount of work on wireless peer based networking. One of the first commercial mesh networks was Metricom's Ricochet network [5] in the mid-90s. Ricochet nodes automatically routed client traffic through half-duplex wireless hops until reaching a hardline connection.

When the 802.11 standard was ratified in the late-90s, other mesh networks started to emerged. One of these is the MIT Roofnet [6], [7] project where tens of access points with roof mounted antennas formed a mesh around campus. Roofnet's emphasis is more on route maintainability and optimization than on handing off a client's connection. Many other community and commercial mesh network implementations also exist, such as Rice University TAPS in Houston [8] and Urbana-Champaign Community Wireless Project [9].

Microsoft Research has also done notable work in the area of mesh networks. Their Mesh Connectivity Layer (MCL) [10] creates a wireless mesh network between Windows clients. Their approach focuses on efficient routing protocols along with the unique support for multiple radios on each node. Adya, Bahl, Wolman, and Zhou have shown [11] that using multiple radios on a mesh node combined with smart routing algorithms [12] will dramatically improve the throughput of a wireless mesh network. Their work necessitates a specific network driver on all mesh network participants, including the clients. Our approach requires no such modification to clients, and works across a variety of operating

systems.

The IEEE 802.11s Mesh Networking standard, analyzed by Camp and Knightly in [13], specifies three different types of mesh nodes. Mesh points (MP) includes all mesh nodes that participate in the wireless backbone to increase the mesh connectivity. Some mesh points serve as mesh access points (MAP), providing connectivity to clients within their wireless coverage area. Also, some mesh nodes may serve as mesh portals (MPP), connecting the wireless mesh to an external network such as the Internet. In our approach, we assume that every node is potentially an access point, as it increases the availability of the system. Furthermore, other than Internet connectivity, we make no distinction between the capabilities available in nodes that are simply MAP, MPP, or both.

1.2.2 Intra-domain Handoff

Cell networks achieve smooth handoff by sharing information between towers about a given mobile device. This session data is used for routing and is updated whenever a phone switches cells [14], [15]. The 802.11 standard lacks the handoff mechanisms available in today's cell network protocols.

Mishra, Shin, and Arbaugh [16] analyzed the link-level handoff performance in current 802.11 hardware. Approximately 90% of a handoff delay is attributable to the client adapter scanning for its next AP. Their experiments also illustrate that the practical handoff delay can vary widely depending on the vendors used for the client network card and the AP. Vatn [17] investigated the latency effects of a wireless handoff on voice traffic. His conclusions echo those of Shin and Arbaugh in that the handoff latency can vary widely depending on the hardware vendor used. Since our approach does not require reassociation during handoff, we do not suffer from these vendor specific delays.

Ramani and Savage [18] recently demonstrated that a quick link-level handoff is possible on 802.11 networks when the client monitors the signal quality of access points and uses a fast scanning mechanism to listen to all APs in range to choose the best one. Their SyncScan system has achieved an impressive handoff as low as 5 ms. The fast scanning is achieved through driver modifications to a client's network adapter. In the contrary, our approach uses any unmodified 802.11 client.

Two well known general approaches to intra-domain handoff are Cellular IP [19] and Hawaii [20]. A comparison is presented in [21]. In Hawaii, or Handoff-Aware Wireless Access Internet Infrastructure, messages are exchanged between the old gateway and the new gateway for forwarding packets. Cellular IP establishes routes based on traffic from the client, and handoff takes place when a cross-over router is reached. However, applications like Push-to-Talk [22] may require packets to be sent to mobile clients that are only receiving traffic. In addition, these approaches rely on clients initiating the handoff process, and do not address the link level handoff delay present in 802.11 networks when clients reassociates with another access point. Other approaches to intra-domain handoff, such as TMIP [23] and [24], improve handoff latency in 802.11 networks but do not overcome these limitations. Other general approaches such as IDMP [25], SMIP [26], and HMIP [27] focus on hierarchy to reduce the global signaling load to improve scalability. In contrast, we provide a complete link-level and network-level solution and propose a novel approach for controlling the handoff from the infrastructure.

In [28], Caceres and Padmanabhan propose the use of gratuitous ARP messages to achieve transparency in the wired infrastructure during handoffs. In their approach, mobile clients initiate the handoff themselves, and the access points send gratuitous ARPs to their upstream routers to create the illusion that mobile clients are always connected to the wired

network. The approach requires all access points to be directly connected to the same wired ethernet network.

Seshan, Balakrishnan, and Katz used a multicast approach in the Daedalus project [29] to ensure timely delivery of client traffic during a handoff in a cell-based wireless computer network available in 1996. Their handoff implementation resulted in a delay as low as 8-15 ms without any lost packets on a 2 Mbps link. In Daedalus, each base station was connected to the same Ethernet network. A non-primary base station near a client would join a multicast group unique to the client to ensure that it could immediately begin forwarding packets if it became the primary serving base station. In contrast to our approach, handoff in Daedalus was initiated by the client upon receiving a stronger signal from a new base station.

Helmy, Jaseemuddin, and Bhaskara show in [30] how fast handoff can be achieved in wireless networks by requiring mobile clients to explicitly join a multicast group to which packets are multicast-tunneled through the infrastructure. Multicast during handoff, refered to as simulcast, is also used during handoff in S-MIP [26]. In a different approach, Forte and Schulzrinne [31] propose a scheme where clients collaborate in multicast groups with each other clients in their vicinity to share useful information about the network and improve handoff performance. Our approach does not require any modifications to the mobile client thus supporting standard mobile devices of any architecture or operating system.

The IEEE has also been working on standardizing handover for wireless IP networks at two different levels. The 802.11r standard aims at providing fast Basic Service Set (BSS) transition by allowing clients to use their current access point as a conduit to other access points. The 802.21 standard aims at providing handover between different network types, commonly known as media independent or vertical handover. These approaches require

modifications to the 802.11 standard, and so to the access points and to every client device. In our approach, no modifications are necessary.

Existing experimental wireless mesh testbeds that support client mobility include MeshCluster [32] and iMesh [33], both of which work with mobile clients in infrastructure mode. MeshCluster, which uses MIP for intra-domain handoff, shows a latency of about 700 ms due to the delay incurred during access point re-association and MIP registration. iMesh also offers intra-domain handoff using regular route updates or Mobile IP. Using layer-2 handoff triggers (no moving client), handoff latency in iMesh takes 50-100 ms. The approach was later used in a more realistic environment for improving VoIP performance in mesh networks, with similar results [34]. SMesh [35, 36] provides 802.11 link-layer and network-layer fast handoff by working in ad-hoc (IBSS) mode, controlling handoff from the mesh infrastructure, and using multicast to send data through multiple paths to the mobile client to deal with incomplete knowledge and unpredictable moving patterns.

1.2.3 Inter-domain Handoff

Two general approaches for supporting inter-domain handoff are Mobile IP (MIP) [37] and Mobile NAT [38]. In MIP, a client binds to an IP address at the Home Agent (HA). As the mobile client moves to a different access point or domain, it receives a Care-of-Address (CoA) from a Foreign Agent (FA). The mobile client then registers its new CoA with its HA, and data is then tunneled through the HA. Our approach does not require binding the mobile client to a specific Home Agent, but rather ties each connection to the Internet gateway that is closest at the time the connection is initiated.

In Mobile NAT, a client receives two IP addresses through DHCP: a binding address for the network stack, and a routing address that will be visible in the network. As the

mobile client moves to a different domain, the client may receive a new routing address. However, as end-to-end connections were initiated from the IP address of the network stack, which remains the same, existing connections will be maintained. The approach requires modifying the mobile client network stack to be aware of the protocol, and also changes in the standard DHCP protocol. Our approach does not require any modifications to the mobile client or the DHCP standard.

Many reactive approaches have been proposed to address Internet connectivity in wireless ad-hoc networks [39–43]. Some of them provide good connectivity while paying the cost of a fairly high overhead due to periodically advertisements from Foreign Agents, while others adjust slower, using a reactive approach and broadcast advertisements to find Foreign Agents on demand. A hybrid approach that achieves the same connectivity as in pro-active protocols but with less overhead was proposed in [44]. These schemes usually share similarities with Mobile-IP and although they are suitable for ad-hoc networks, they do not perform well in wireless mesh networks. Backbone nodes in a mesh network are stationary, as opposed to the nodes in ad-hoc networks, leaving space to more efficient protocols that exploit the relative stability of the mesh nodes.

Our work also relates to hybrid networks that connect some of the nodes through the wired network to improve efficiency in the use of the wireless spectrum [45]. An interesting problem addressed in [46–49] deals with interconnecting wireless LANs with cellular networks. This problem is complementary to our work, which focuses on interconnecting wired and wireless networks.

1.2.4 Overlay Networks

Overlay networks enable developers to implement new services on top of the IP network infrastructure without requiring special support from the underlying network. They are usually built as application level routers to ensure flexibility and usability across platforms, at the cost of requiring packet to traverse through user space. Examples of application level overlay routers include RON [50], End-System-Multicast [51], and Spines [2, 52].

RON routes packets through a user level router on an overlay network to increase the reliability of the end-to-end path when compared to using the underlying direct path. End-System-Multicast also routes through an application router to support overlay multicast without infrastructure support.

Spines is a more generic overlay network that provides transparent multi-hop unicast, multicast and anycast communication with a variety of link and end-to-end protocols. For example, semi-reliable links can recover from some loss in the overlay links while packets are independently forwarded to their destination in order to improve VoIP [53] quality. Spines has a socket-like interface that makes the interconnection with other components very easy. It uses an addressing space composed of virtual IP addresses and virtual ports. Regular socket calls such as *sendto()* or *recvfrom()* are mapped directly into Spines API calls. The SMesh system presented in this thesis instantiates a Spines daemon on each wireless mesh node to manage group membership and to forward messages within a multi-homed wireless mesh network.

Chapter 2

SMesh, A Seamless Wireless Mesh Network

In this chapter we present the wireless mesh network paradigm and introduce our wireless mesh network system, SMesh [1], that we developed to realize the protocols and algorithms presented in this thesis.

We first generalize the mesh networks paradigm, and show the inherent hierarchy in these networks where two classes of participants, mesh nodes and mesh clients, participate in different capacity: mesh nodes communicate with each other, possibly using multiple hops, while mesh clients connect directly through a mesh node, each of which serves as an access point. This is one of the main differentiating factors between the mesh network and the mobile ad-hoc network paradigm, where everyone (mesh nodes and mesh clients) participate as equal in the overall routing strategy. We then introduce our architecture which manages the clients through an overlay network in the mesh. Finally, we show how our architecture overcomes a system limitation to divert packets to user space and how we use this to communicate through an overlay network.

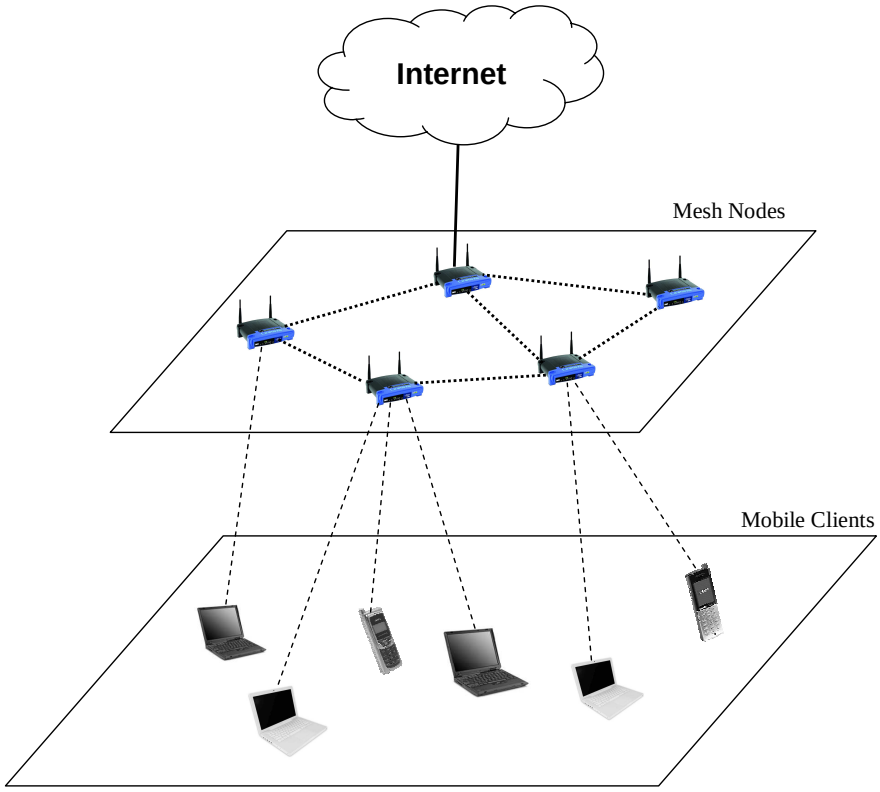


Figure 2.1: A two-tier wireless mesh network

2.1 Wireless Mesh Networks

Wireless mesh networks provide a promising paradigm to increase the mobility range of wireless devices. In these networks, multiple access points create a mesh topology and forward packets using multiple wireless hops. Some of the access points in a mesh network may be connected to the Internet, while others may not. Mesh clients connect to the mesh through one of these access points. Figure 2.1 depicts a general overview of a the wireless mesh network paradigm.

While the access points of a mesh network are usually stationary, mobile devices that connect to the mesh network can roam throughout the coverage area and may require con-

tinuous service for peer-to-peer communication as well as for external Internet connectivity.

Mesh networks are usually self-organizing and easily deployable. They are useful for providing connectivity in remote geographical areas, as well as for first responders at disaster affected locations that lack the wired infrastructure. In such scenarios, providing support for real-time applications such as VoIP is often critical.

2.2 The SMesh Architecture

We consider a set of stationary 802.11 access points connected in a mesh network, and a set of wireless mobile clients that can move within the area covered by the access points. We call each access point a *node* in the wireless mesh network.

The mesh topology changes when wireless connectivity between the mesh access points changes, when nodes crash or recover, or when additional nodes are added to expand the wireless coverage. Mobile clients are not part of the mesh topology. Some of the mesh nodes, but not all, have a wired Internet connection. We refer to them as *Internet gateways*. Each mesh node should be capable of reaching its closest *Internet gateway* or any other node via a sequence of hops.

The mobile clients are unmodified, regular 802.11 devices that communicate with the mesh nodes to get access to the network. We do not assume any specific drivers, hardware, or software present on the clients. Therefore, *any* regular unmodified mobile device should be able to use the mesh network transparently.

Our goal is to allow mobile clients to freely roam within the area covered by the wireless mesh nodes, with no interruption in their Internet connectivity. All connections (reliable or best effort) opened at mobile clients should not be affected as the clients move throughout

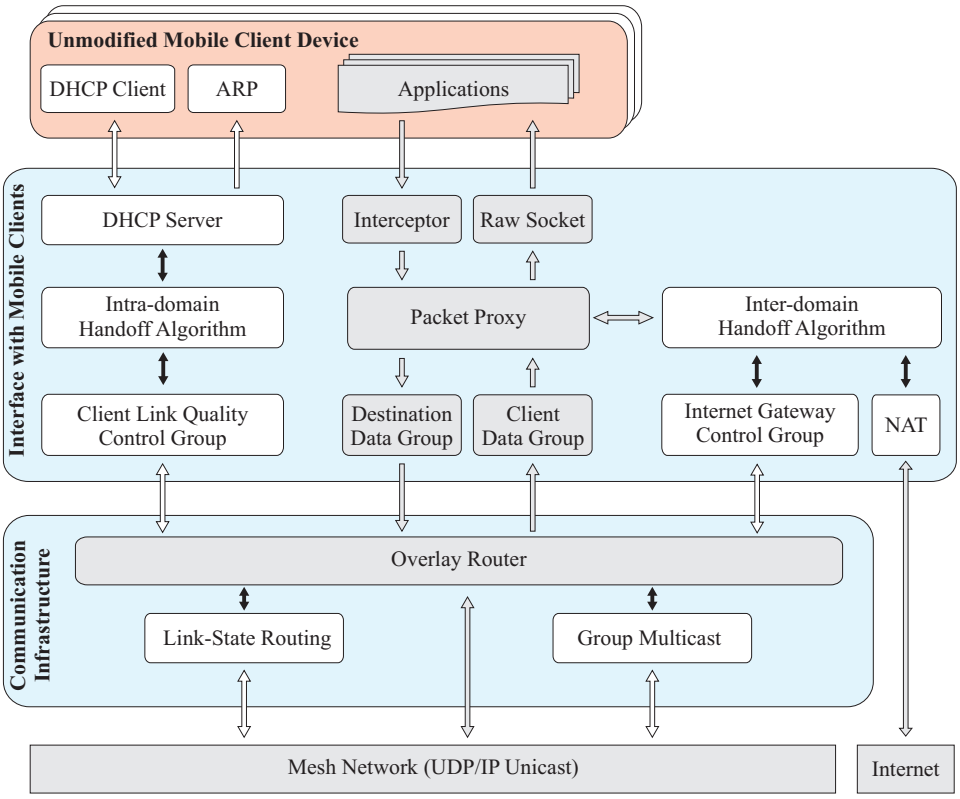


Figure 2.2: The SMesh Architecture

the coverage area served by the wireless mesh.

Following the above goals, we implemented SMesh [35, 36], a system that is capable of providing seamless wireless connectivity to mobile clients. The software architecture of SMesh is shown in Figure 2.2. Below we describe the two main components of the SMesh architecture: the communication infrastructure and the interface with mobile clients.

2.2.1 Overlay Communication Infrastructure

The mesh nodes create a relatively stable ad-hoc wireless network. Within this network, the nodes need to forward packets over multiple hops in order to communicate with each

other for reaching the Internet gateways or for coordinating decisions about serving mobile clients. The nodes also need to discover and monitor their neighbors and to automatically adjust the mesh routing in case of topology changes.

The communication infrastructure in SMesh is based on the Spines messaging system [2, 52]. The Spines overlay network interconnects all nodes through direct links in the wireless network and through virtual links in the wired network. SMesh instantiates a Spines daemon on each wireless mesh node to forward messages within the wireless mesh. Each daemon keeps track of its own direct neighbors by sending out periodic hello messages. Based on the available connectivity, each node creates logical wireless links with its direct neighbors and uses a link-state protocol to exchange routing information with other nodes in the network.

The nodes flood link-state information using reliable links between direct neighbors. This allows the nodes to send only incremental updates, and only when network topology changes. Link state updates contain only information about the wireless links that change their status. When there are no changes in topology, no routing information is exchanged. Considering that mesh nodes (access points) are mostly stationary and that topology changes are relatively rare, the incremental link-state mechanism incurs very low overhead. Note that in SMesh, mobile clients are not part of the mesh topology.

While this link-state protocol may not be optimal for a general ad-hoc network, it is optimized for the relatively stable network underlying our mesh of access points.

Spines allows us to use multicast and anycast functionality in a multi-hop wireless environment without infrastructure support. A multicast group is defined as a class D IP multicast address while an anycast group is a class E IP address. Note that the groups are defined in the Spines virtual addressing space, not in the actual IP address space of the

network. When a mesh node joins or leaves a group, the local Spines daemon informs all the other nodes in the network through a reliable flood similar to the link-state protocol. Only joins and leaves are flooded to the mesh nodes in the system. The group membership is maintained in Spines in tuples of the form (*mesh_node_address, group_address*), such that each node knows all the groups that other nodes are members of.

Based on the group membership and available connectivity, Spines automatically builds multicast trees throughout the mesh network. A multicast data message follows the multicast tree corresponding to its group. Therefore, if several nodes in a certain vicinity join a multicast group, multicast messages exchanged between them will only be sent in that vicinity. An anycast data message follows a single path in the tree to the closest member of the group.

Multicast trees in Spines are built by optimizing on a metric that can be related to the number of hops, link latency or loss rate. In our tests, Spines could handle several hundred thousand group members on regular desktop machines and was limited only by the available memory to maintain the data structures. SMesh instantiates two groups for each client, with a few members in each group. The more limited Linksys WRT54G routers used in our experiments have enough memory to support at least 1000 mobile clients at the same time.

2.2.2 Interface with Mobile Clients

SMesh provides the illusion of a single distributed access point to mobile clients. This is achieved by providing connectivity information to clients through DHCP [54], and by routing client packet through the overlay network.

2.2.2.1 Mobile Client Connectivity

The DHCP Server running at each mesh node (access point) is in charge of providing network bootstrap information, including a unique IP address, to a requesting client. We compute this IP address using a hash function on the client's MAC address, mapped to a class A private address of the form 10.A.B.C. A small portion of the private IP addresses in this range is reserved for SMesh nodes, and the rest are available to mobile clients. In case of a hash collision, the client with the smallest MAC keeps the current IP and any other client in the collision gets a managed IP. This scheme decreases the amount of IP management in the network, while assuring that each client gets the same IP address from any SMesh node.

Of particular importance in the DHCP protocol are the *Server ID*, *Default Gateway*, and the T_1, T_2 and *Lease* timers. The *Default Gateway* specifies the next hop router to use at the MAC level when sending to an IP address outside the client's netmask. The *Server ID* specifies the DHCP Server IP address that the client should contact to renew its lease. The T_1 and T_2 timers specify when to start unicasting or broadcasting DHCP requests (`DHCPREQUEST`), and the *Lease* timer specifies when the client must release the IP address. After the *Lease* timer expires, all the connections at the client are terminated. If the access point responds to a DHCP request before the client's Lease time expires, it is able to keep all connections open. In SMesh, the lease time is set to 90 seconds, which gives a client enough time to reconnect in case it goes out of range of any of the mesh nodes temporarily.

Table 2.1 shows our addressing scheme. We set the netmask of the client to a very small network, thus forcing the client to send packets destined to the Internet or a peer through its

Type	Address	Example	Detail
Client IP	10.A.B.C	10.11.12.25	Assigned by SMesh DHCP Server
Netmask	255.255.255.248	255.255.255.248	Assigned by SMesh DHCP server
Default Gateway	10.A.B.C + 1	10.11.12.26	Assigned by SMesh DHCP Server
Network Address	10.A.B.C - 1	10.11.12.24	Calculated by Client with Netmask
Broadcast Address	10.A.B.C + 6	10.11.12.31	Calculated by Client with Netmask
Reachable IP	10.A.B.C + 2	10.11.12.27	Used by SMesh for monitoring client

Table 2.1: SMesh IP address assignment scheme

default gateway. The default gateway is a virtual IP address; there is no node in SMesh with that IP address. Instead, SMesh makes the client ”believe” that this address is reachable by associating this IP address to a mesh node hardware address. This forces the client to route packet through SMesh.

While each client in SMesh consumes 3 bits from the address space, there are still 21 bits available, which allows us to support over one million client IP addresses.

We will explain in Chapter 3 how the default gateway is mapped to an access point, how we use the different DHCP timers, and how the additional IP address in the client network is used for monitoring the client. The handoff algorithms will be explained in Chapters 3 and 4.

2.2.2.2 Packet Proxy

Mesh nodes serve as default gateways for the mobile clients. A Packet Proxy module, depicted in Figure 2.2, uses an interceptor to grab packets from a client, and a raw socket interface to forward packets back to the client. The interceptor is explained in detail on Section 2.2.2.3.

Each mobile client is associated with a unique multicast group to receive data (Client

Data Group). One or more mesh nodes that are in the vicinity of a client will join that client's Data Group. All the Internet gateway nodes are members of a single anycast group.

If the destination of a packet is a SMesh client, the packet is sent to the SMesh nodes that joined that client's Data Group. The mesh node sending this packet can be the Internet Gateway (for packets coming from the Internet) or a sending client access point (for packets originated by a different SMesh client). Upon receiving a packet for the client, each of the SMesh nodes that joined that client's Data Group forwards the packet to the client.

If the destination of a packet is the Internet, then the packet is sent by the originating client's access point to the closest Internet gateway by forwarding it to the anycast group. The Internet Gateway will then forward the original packet to the Internet using Network Address Translation (NAT) [55]. When a response packet is received from the Internet, a reverse NAT is performed and the packet is sent to the appropriate Client Data Group.

Spines forwards the packets to the members of the client's Data Group using a multicast tree. This way, if the mobile client moved, and a different SMesh node joins the client's Data Group, the packets are forwarded to the newly joined SMesh node. The SMesh node(s) in the Client Data Group use a raw socket to deliver the packet, allowing the mobile client to receive the packets unmodified as if it had a direct connection to the end host. If there are multiple nodes in the Client Data Group, the client could receive duplicate IP packets. However, duplicate IP packets are dropped gracefully at the receiver (TCP duplicates are dropped at the transport level, and applications using UDP are supposed to handle duplicates).

2.2.2.3 Transparent Overlay Proxy

Application level overlay networks forward packets through application level routers, thus requiring packets to traverse user space. RON used this approach with a special divert socket to increase resilience in the Internet.

SMesh intercepts clients packets and sends them through the Spines overlay network to the access points serving the destination. The overlay may span wireless and wired links, and routes may take advantage of the wired network to optimize wireless usage. Once the packets are received by the destination's access points, SMesh strips the overlay headers and forwards the original packet to the mobile client using a raw socket. Unlike RON, our interceptor relies only on a packet sniffer socket, which is readily available in most operating systems, as well as filter and firewall settings, to perform this task.

In our approach, we use the libpcap library [56], a well known application level interface for user-level packet capturing. In addition, to improve performance, we use Berkeley Packet Filters [57] to ignore unwanted packets in the kernel. The mesh nodes configure each node as follows:

- Disable packet forwarding so that the overlay is the only one forwarding packets in the mesh network
- Drop any packet destined to the Internet IP address of mesh nodes connected to the Internet.
- Filter out every port used by the overlay network to ensure that these packets are not captured. Spines uses four different ports to communicate between daemons.

When a mesh node receives a packet destined to an IP address that is not its own (i.e., when a mobile client sends a packet destined to the IP address of Goggle), the kernel

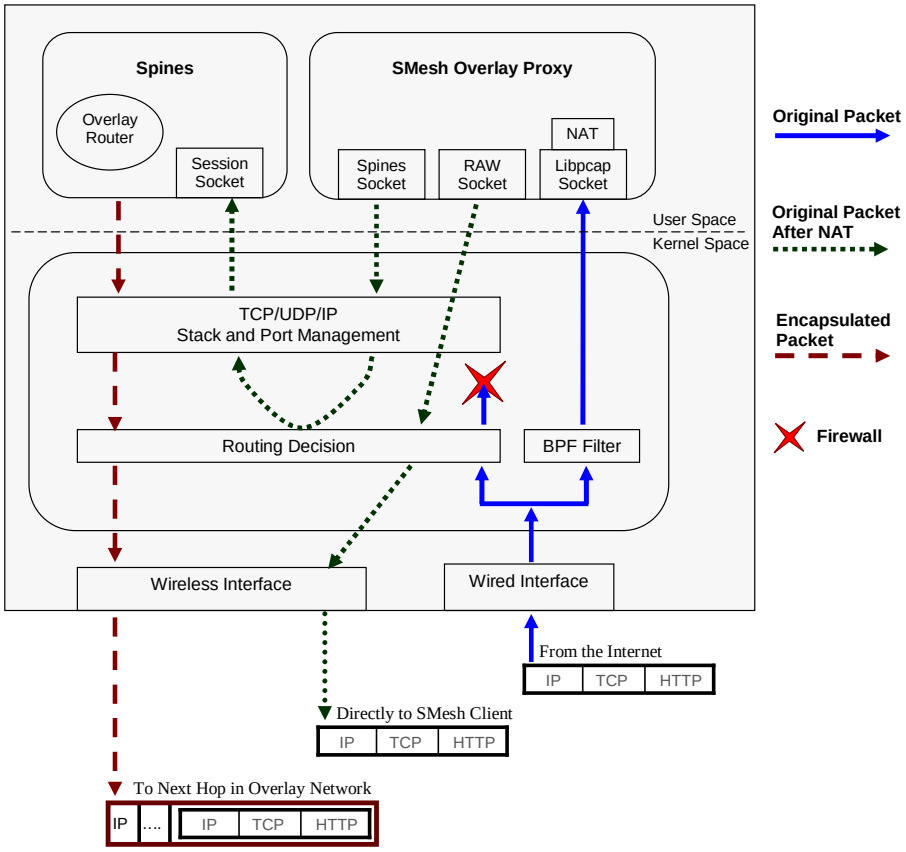


Figure 2.3: SMesh Transparent Overlay Proxy with a packet flowing from the Internet to a mesh client

attempts to route the packet, and when unsuccessful it drops the packet to the floor. However, the packet sniffer socket gets a copy of the packet, which SMesh then send through the Spines overlay network to its appropriate destination. As previously explained, when a packet reaches the SMesh Internet Gateway, a Network Address Translation is performed. Every packet coming back from the Internet will have the Internet gateway as the destination IP address. To ensure that this mesh node does not act on these packets (e.g., by reseting a TCP connection that it did not start), a firewall needs to be enabled to drop any packet destined to this address. At each end point of the overlay network, a raw socket allows us to send the exact packet to its destination, effectively creating a transparent tunnel

through our overlay in the wireless mesh network.

Figure 2.3 shows the different components that allow us to intercept packets, and how a packet flows from the Internet to a mesh client. In this case, the mesh Internet gateway is handling the client, so it forwards the packet directly to the Client. It also forwards the packet to Spines, who will forward the packet to any other Spines daemon in the Overlay Network who has a member in the Data Group for that client. If there is no other member, Spines will simply drop the packet. As we will see in the next chapter, it is possible for more than one access point to be a member of the client Data Group.

Chapter 3

Achieving Fast Intra-domain Handoff

Real-time applications such as VoIP require that packets arrive on a steady stream. Any burst of loss where consecutive packets are lost results in degradation of quality. In addition, packets should arrive within 100ms to prevent a noticeable delay that impairs interactivity, and delay variability should stay below 20ms to ensure the highest quality of service. Therefore, a handoff protocol should be fast enough to avoid any packet loss, and should ensure that packets are delivered to their destination in a timely manner.

In this chapter we present our fast intra-domain handoff protocol for wireless mesh networks. We first describe the problem that current 802.11 networks face when a handoff is required between access points. We then describe how we monitor the client, and how we asses the quality of the link to the client from that a mesh node. Then, we present our approach to fast intra-domain handoff, and finalize by demonstrating the performance of our fast handoff protocols in a testbed consisting of 15 mesh nodes.

3.1 Motivation

When 802.11 devices are configured in *infrastructure mode* (BSS), they inherently perform their own scanning for a better access point. A layer 2 handoff takes place through a re-association request/response process which can last as long as several seconds [58]. In addition, this handoff is both hard and forward; hard because the client can only speak with one access point at a time, and forward because the client can not communicate with it's old access point during the handoff process. A typical handoff will last about five-hundred milliseconds, which translates to dozens of lost packets during handoff for VoIP applications.

In order to avoid this behavior and control the handoff solely from the access points, we configure both the access points and the mobile clients in *ad-hoc mode* (IBSS). This setting is part of the normal setup of any 802.11 device.

One way to perform the handoff in ad-hoc mode is by relying on the DHCP protocol. Given that a DHCP request is broadcasted by the client after T_2 seconds (Rebind timer) a different access point is allowed to respond and become the default gateway for the client. Even if T_1 (Renew) and T_2 timers are set to very small values (e.g., 2 seconds), handoff can still take seconds. Moreover, because the first DHCP response is considered, the client may connect through an access point that has a weak connection, while better nodes may be available. A handoff of a few seconds may seriously affect some applications such as VoIP, which require packets to arrive within a limited time, as low as 100 ms, before being considered lost.

Instead of letting the client "decide" when the handoff should take place by relying on the DHCP protocol, we make the SMesh nodes track their connectivity to the client and force the client to change its access point when better connectivity is available (avoiding

oscillations is described below). To achieve this without modifying anything on the client side, we provide the illusion of a single IP that never changes as the default gateway of the client and use gratuitous ARP messages to force roaming to the SMesh node with the best client connectivity.

The details of our handoff protocol are described below. These include the link quality metric used by SMesh to determine the best access point for each client, the use of overlay multicast groups for managing the clients, and the actual handoff process.

3.2 Mobile Client Monitoring

3.2.1 Seamless Heartbeat with DHCP and ARP

SMesh provides the illusion of a single distributed access point to mobile clients. This is achieved by providing connectivity information to clients through DHCP, by always giving the same information (IP address, Netmask, and Default Gateway) to the mobile client, and by routing packets through the wireless mesh network.

In order to provide continued connectivity and availability to the mobile client, we need to continuously monitor the client. To achieve seamless monitoring without any involvement from the client, we developed two strategies.

1. DHCP (Dynamic Host Configuration Protocol)

   According to the DHCP standard [54], the T_1 (Renew) and T_2 (Rebind) timers specify when to start unicasting and broadcasting, respectively, DHCP requests (`DHCPREQUEST`), and the *Lease* timer specifies when the client must release the IP address. After the *Lease* timer expires, all the connections at the client are ter-

minated. If the access point responds to a DHCP request before the client's Lease time expires, it is able to keep all connections open. When using the SMesh DHCP monitor, our DHCP server instructs the clients to renew their IP address every 2 seconds, thus serving as a heartbit to keep track of the client. In addition, the timers may be set so that the client unicast or broadcast their request every 2 seconds. On the down side, it employs a non-negligible overhead as a `DHCPREQUEST` packet is at least 300 bytes long, and a `DHCPACK` is about 548 bytes. This is the approach we took in [35].

2. ARP (Address Resolution Protocol)

   ARP [59] protocol is used to map an IP address to a hardware address (MAC), when a host (or router) wants to communicate with another host inside the same network. However, even if the hardware address is known, we can still use this protocol to probe the client's link and estimate its loss rate. By using regular ARP requests, we can make the client either unicast or broadcast ARP responses. We instruct the client to respond to the IP address available in it's own network, and the MAC address of the SMesh node that sent the ARP request. This is necessary as the real IP addresses of the SMesh nodes is outside the client network. Also, to limit the number of access points probing the client, only the one in the client *Data Group* periodically sends a request, and all nodes in the vicinity use the reply to compute the metric. If a node stops hearing the replies, it attempts to probe the client at least once. The advantage of using this approach is that, unlike DHCP, ARP packets are very small, only 28 bytes. In SMesh, we request an ARP reply from the client every one or two seconds. This is the approach that we take in our updated version of SMesh and for the experiments presented in this thesis.

It is also possible to use regular packets sent by the client to monitor its connectivity, which happens when the client is sending or receiving [1] packets. However, when a client is idle, and traffic needs to be sent to it's current location, we either need to know the routes immediately by one of the methods described, or a paging mechanism [60] is necessary to allows us to find the client within some reasonable time. We proactively monitor the client to ensure that routes are immediately available, which allows us to support applications like Push-To-Talk [22] that may require data to be sent to a mobile client that is not sending or receiving data at that specific point in time.

3.2.2 Quality Metric

We use the monitoring schemes described above to keep track of the quality of the links to mobile clients. Both schemes allow us to receive either unicast or broadcast replies from the client. Using broadcast instead of unicast eliminates the MAC level retransmissions of requests, which allows us to estimate more accurate the loss rate.

Each SMesh node computes a client link quality metric based on the observed loss of a client's DHCP requests or ARP responses, using the following weighted average decay function:

$$M_{new} = M_{old} * D_f + Current * (1 - D_f) \ , \ 0 < D_f < 1$$

where M is the link quality measure and D_f is the decay factor. $Current$ is a constant value which is set to 0 if the access point did not receive any DHCP or ARP probe packets responses in the expected time, or is set to a maximum value if a probe packet is received. The access point calculates this function every second for each client in its vicinity. SMesh

[1] When a client is receiving data, it needs to send an acknowledgement at the 802.11 level for every packet it receives, which can also be used to monitor connectivity.

uses a decay factor of 0.8 to make the protocol resilient to occasional wireless losses of the probe packets, while maintaining its adaptability to network condition changes. SMesh uses a $Current$ value of 50 to allow integer calculations with discrete mapping. The tie breaker between two access points having the same integer metric (in the range of 0 to 50) is according to the lowest IP of the access point.

Many wireless devices allow applications to capture packets through a monitoring interface. When the mesh node is also equipped with such an interface (as in the case of our Linksys routers), specific radio measurements from the received packet, as well as the complete 802.11 frame, is available to SMesh, as follows:

1. RSSI (Received Signal Strength Indicator) RSSI is a measurement of the radio signal strength. If the wireless interface is configured in monitor mode, an additional header is added by the wireless driver, which contains the RSSI information. One thing we must be aware of is that the RSSI value must be in the same range of values for all mesh nodes. If different card manufacturers are used, a conversion might need to be performed (e.g., Cisco Systems cards report a maximum RSSI value of 100, while Atheros cards report a maximum of 60).

2. 802.11 Retransmission Flag Every unicast packet transmitted in 802.11 needs to be acknowledge by the recipient. If the packet or the acknowledgement is lost, the sender retransmits the packet, and sets a retransmit flag in the 802.11 header. The maximum number of retransmissions is usually four. In our case, instead of having to make the client broadcast to know when packets are lost on the first transmission, we look at this flag to determine if the packet was lost on the first attempt.

The main advantage of using RSSI versus a loss-rate only measurement is that we can start the handoff process to a better access point before there is any loss in the medium.

The initial loss in the medium is usually masked by the 802.11 retransmissions, so the client sees this loss as an increase in latency for these packets. However, RSSI alone is not a good indication of the loss rate of a link, so we use it in conjunction with the loss rate, adjusted with the decay function described above, for measuring the quality of the link.

3.3 Intra-domain Handoff Management

3.3.1 Mobile Client Data Group

A mesh node joins the client Data Group so that it can receive and forward data packets for that client, if it believes it has the best connectivity to the client based on link quality metrics it receives from other nodes in the client's Control Group.

Nodes in a Client Data Group receive data packets that need to be forwarded to the group's corresponding mobile client. If more than one node is a member of a client's Data Group, duplicate packets will be sent to that client by each member of that client's Data Group.

Our protocol must guarantee that, at all times, there is at least one member in the Data Group of each client, such that the client will be served by at least one mesh node. On the other hand, it would be wasteful to allow more than one node in the vicinity of a client (and therefore in the Control Group) to also be in the Data Group most of the time as this creates duplicate packets. Our protocol balances between these two conflicting goals (availability and efficiency).

3.3.2 Mobile Client Control Group

In addition to the previously described Client Data Group, used for forwarding data packets in SMesh towards access points serving the client, the access points in the vicinity of a client join a different multicast group specific to that client, called Client Control Group. The Client Control Group is used to coordinate with other mesh nodes in the client's vicinity regarding link quality metrics and regarding which access point will be the best to serve that client. A mesh node joins a client's Control Group when it receives one of the heartbeats from the client, and leaves the client's Control Group after not hearing from the client for some time. For example, for a mobile client with address 10.A.B.C, a SMesh node will join the client's Control Group at 224.A.B.C and, if needed, the client's Data Group at 225.A.B.C. This maps every client to a set of two unique multicast groups[2].

The link quality metric is shared by the access points periodically by posting it on the client's Control Group. Since only the nodes receiving a heartbeat from a client join the client's Control Group, the multicast overhead is localized only in the vicinity of that client and will not propagate beyond that in the network.

3.3.3 Client Handoff

Each mesh node has its own IP address that allows it to communicate with other mesh nodes. However, in order to provide a completely transparent handoff to clients, mesh nodes advertise a single virtual gateway IP address to all clients in their DHCP offers and acknowledgements (`DHCPOFFER` and `DHCPACK`). Mobile clients set their default gateway to this virtual IP address regardless of which access point they are connected to. This way, mobile clients get the illusion of being connected to a single access point that follows them

[2]Control Groups and Data Groups are implemented as Spines multicast groups.

as they move. The IP address of the default gateway only appears in the DHCP offer and in subsequent ARP requests, as described below. In all other IP communication with mobile clients, the default gateway does not even appear in the IP packets. It can be set any valid IP address as the communication with the mobile clients is solely based on MAC addresses.

In general, given an IP address for which its corresponding hardware address is not present in the ARP cache of a client, the ARP module of that client will broadcast an ARP request packet. In addition to the source and destination IP addresses, this ARP request contains the MAC address of the source. The value of the destination MAC is not yet known. All the hosts on the local network receive the packet and compare the destination IP with their own IP address. The host for which the IP address matches will issue an ARP reply, filling in the destination MAC field with its own MAC address. This packet is sent directly via unicast to the requesting client. All other hosts will discard the ARP request.

The SMesh handoff mechanism uses gratuitous ARP messages for instantaneous client handoff. A gratuitous ARP is an ARP reply that is not sent as a reply to an ARP request, but rather is sent to the local network voluntarily. Upon receiving such a packet, a hosts will update its ARP caches with the value it received. Typically, gratuitous ARPs are used by hosts to advertise their new hardware address when their network card is changed.

When a SMesh node believes it has the best connectivity with the client and decides to serve that client, it sends a gratuitous ARP as a unicast, directly to the client, thereby changing the MAC address of its default gateway. Subsequent packets sent by the client will be sent to the new access point, following the new hardware address. All operating systems that we have tested accept gratuitous ARPs and begin using the new MAC-IP mapping immediately.

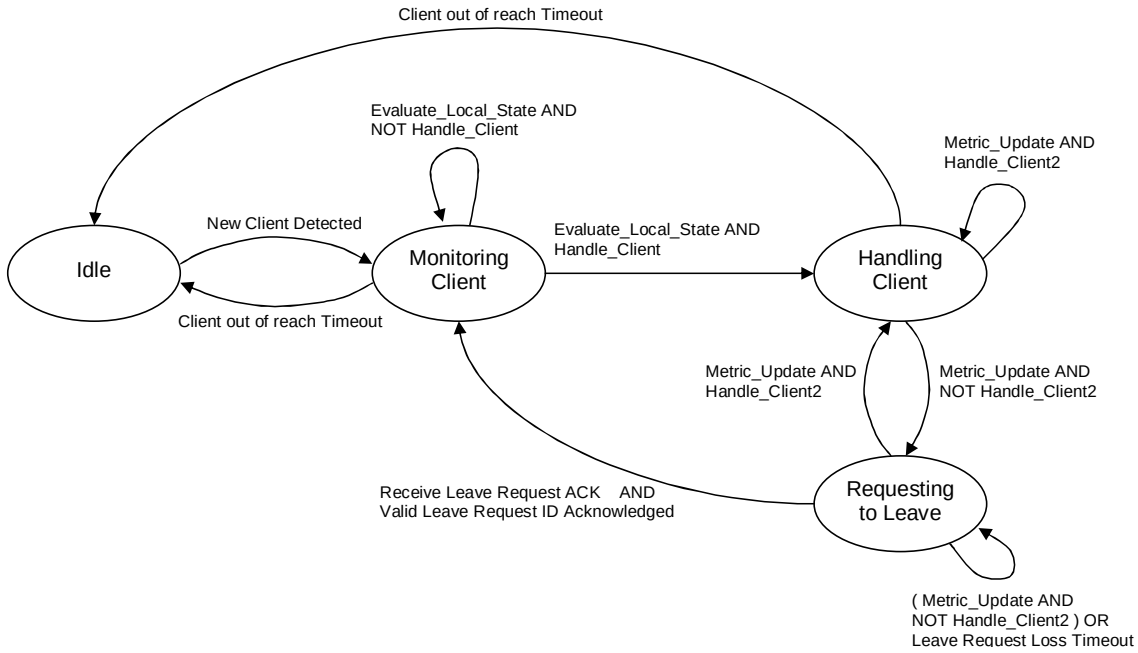
A gratuitous ARP is also sent by an access point when a Leave Request Acknowledge-

ment is sent to another access point, and periodically (e.g., every minute) by the members of the Client Data Group to refresh the ARP entry in the client's ARP table.

In addition to sending a gratuitous ARP to the mobile client, when a node believes it has the best link quality to a mobile client, it joins its *Data Group* so that packets destined to the client start flowing through this access point. If another node is also a member of the Data Group, packets destined to this client are forwarded to both mesh nodes, and each of them forwards the packets directly to the mobile client. The mobile client may receive duplicate packets at this time. Using multicast helps achieve uninterrupted connectivity during handoff by: (1) sending packets through multiple access points to the mobile client, to deal with unexpected client movements while the best access point for the client is chosen, and (2) avoiding loss while route changes take place in the wireless mesh.

A mesh node that joins the Data Group of a mobile client immediately sends a metric update on the Control Group to inform any other node of its latest metric, noting that it is now a member of the client's Data Group. When a mesh node that is a member of the Data Group receives a link quality metric update that shows that a different node in the Data Group is better connected, it issues a *Leave Request*. Leave Requests, sent on the Control Group, are piggy-backed on link quality metric updates. A Leave Request can be acknowledged only by a node in the *Data Group* that believes that it has the best connectivity to the client. A node may leave the Data Group if and only if its request is acknowledged by at least one other node.

The state machine for handling mobile clients is depicted in Figure 3.1, and the pseudocode depicting our algorithm is shown in Figure 3.2. Note that a node checks periodically (line A4) if it should service the client, instead of checking immediately after receiving a metric update, to be less aggressive in taking a decision. However, nodes that are already



Conditions

Handle_Client:	My_Metric > Highest_Metric(Data Group) * Threshold AND My_Rank(Nodes in MonitoringClient state) <= Maximum_Concurrent_Joins (in our case 2)
Handle_Client2:	My_Rank(Nodes in HandlingClient or RequestingToLeave state) == 1
My_Rank(list):	sort list in decreasing order of their metric value, then by IP address to break ties, and return index of local node

Figure 3.1: State Machine for handling mobile clients

servicing the client check their state immediately after receiving an updated metric (line F2) to service the handoff as fast as possible. During disagreements, more than one node may be a member of the Data Group for some time, until the disagreement is resolved.

When a node issues a *Leave Request*, it includes a unique id that increases each time the mesh node enters the RequestingToLeave state (line B11). A node can acknowledge a *Leave Request* only if it is currently the one handling the client (line D2). Note that a node cannot leave unless it receives an acknowledgment with the ID used in the last *Leave Request* (line E2).

```
// Abbreviations: DG = data group, CG = control group, LR = leave request

States = {Idle, MonitoringClient, HandlingClient, RequestingToLeave}
LR_ID = 0

A1. New_Client_Detected(client i):
A2.    Join(CG_i)
A3.    state_i = MonitoringClient
A4.    Periodically(Evaluate_Local_State(i))
A5.    Periodically(Monitor_Client(i))
A6.    Periodically(Send_Metric_Update(CG_i))

B1. Evaluate_Local_State(client i):
B2.    if (state == MonitoringClient)
B3.       My_Rank = Compute_My_Rank(CG_i Members in state == MonitoringClient)
B4.       if (My_Metric_i > (Highest_Metric(DG_i Members) * Threshold) and My_Rank <= 2)
B5.          Join(DG_i)
B6.          Send_Gratuitous_ARP(i)
B7.          state_i = HandlingClient
B8.    else if (state == HandlingClient)
B9.       My_Rank = Compute_My_Rank(DG_i Members)
B10.      if (My_Rank != 1)
B11.         LR_ID_i = LR_ID++
B12.         Send(LR_{LR_ID_i})
B13.         state_i = RequestingToLeave
B14.   else if (state == RequestingToLeave)
B15.      My_Rank = Compute_My_Rank(DG_i Members)
B16.      if (My_Rank == 1)
B17.         state_i = HandlingClient
B18.   if (current_state_i != previous_state_i)
B19.      Send_Metric_Update(CG_i)

C1. Compute_My_Rank(list):
C2.    sorted_list = new list sorted in decreasing order of metric value,
          using node_id to break ties
C3.    return the rank/index where local node is located in sorted list

D1. Receive_LR(client i):
D2.    if (state_i == HandlingClient)
D3.       Send_ACK(LR_i, ID(LR))
D4.       Send_Gratuitous_ARP(i)

E1. Receive_LR_ACK(client i):
E2.    if (state_i == RequestingToLeave and ID(LR_ACK) == LR_ID_i)
E3.       Leave(DG_i)
E4.       state_i = MonitoringClient

F1. Metric_Update(client i):
F2.    if (state == HandlingClient or state == RequestingToLeave)
F3.       Evaluate_Local_State(i)

G1. Client_out_of_reach_timeout(client i):
G2.    if (I_am_member(DG_i))
G3.       Leave(DG_i)
G4.    Leave(CG_i)
G5.    state_i = Idle
```

Figure 3.2: Pseudocode for deciding when to join and leave the Control and Data Groups.

To understand how our algorithm works, let us consider Figure 3.3, where a Client is within the vicinity of 5 mesh nodes. In this example, a handoff is taking from mesh node 4 to mesh node 5. All of the mesh nodes in the vicinity of the client are members of the Control Group for that client, and two of them are also members of the Data Group for that

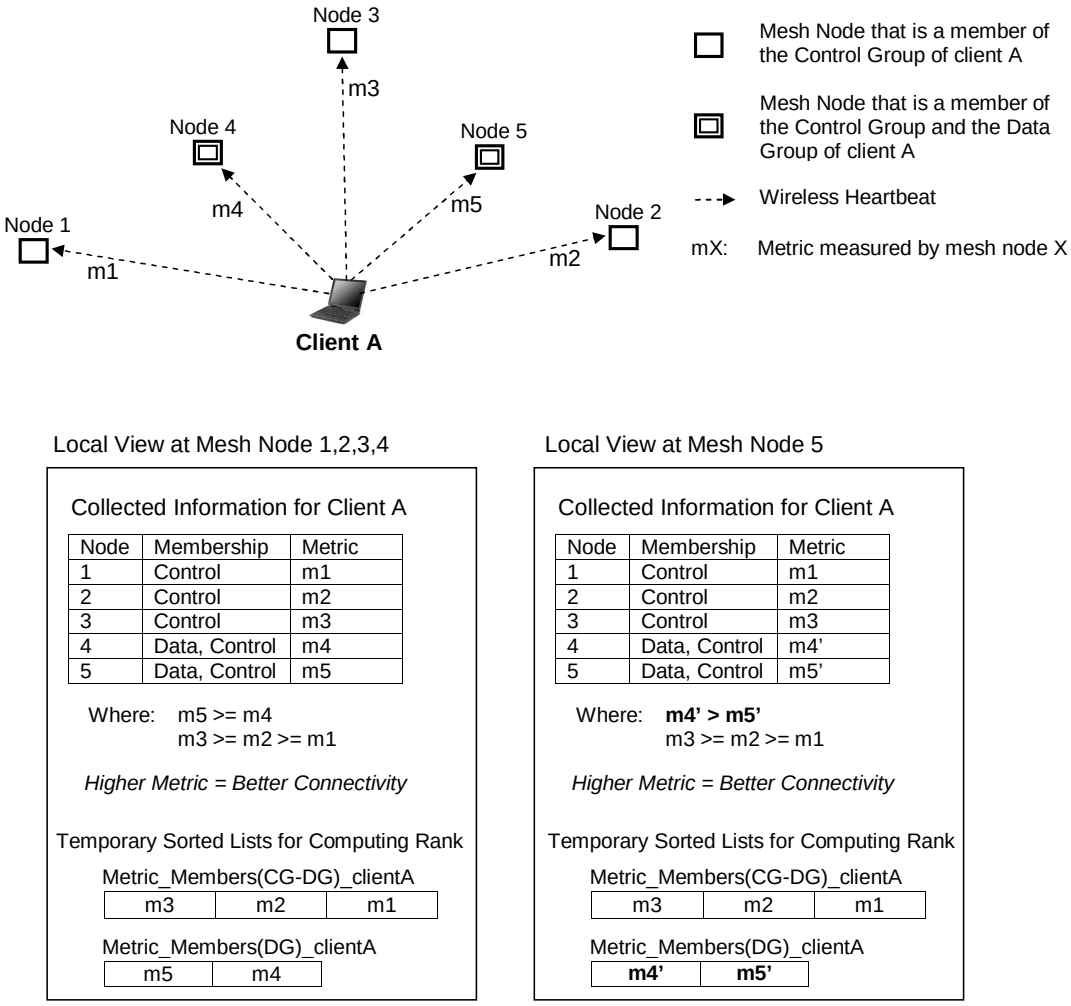


Figure 3.3: Local view of client during handoff based on a distributed monitoring approach

client.

When a node re-evaluates its position about whether to join or leave the data group, it creates two temporary lists, each containing the members of the data or the control group to compute its rank (pseudocode line: C1). A node that is a member of the Data Group is placed only in the Data Group list; other nodes that are members of only the Control Group are placed on the Control Group list. The lists are sorted in decreasing order of metric values, using the last metric received from each of the other nodes. The IP address of a

node is used as a tie breaker. Therefore, the node with the highest metric will be placed at the leftmost position of its list. The local view of a node and the temporary lists that it creates are depicted in Figure 3.3. Each node, after computing its temporary lists, will make a decision as follows:

Node 1: This node is a member of the Control Group only, and should consider joining the Data Group if its metric is bigger (above some threshold) than the metric of the node in the first position in the Data Group list.

Node 2: This node is a member of the Control Group only, and should also consider joining the Data Group. The reason is that this node is not aware of the local view of Node 1 (i.e., Node 1 may think that Node 2 is in the first position). A node in this position will join the data group if its metric is bigger (above some threshold) than the metric of the member in the first position of the Data Group.

Node 3: This node is a member of the Control Group only, and should not consider joining the data group. The reason is that we want to contain the number of nodes that can suddenly join the data group for a node to limit the overhead associated with membership changes and maintain some stability during handoff. No action is taken.

Node 4: This node is a member of the Data Group, and from its point of view, it is not the number best node in his group. This node will send a Leave Request and continue to service the client until it receives an acknowledgment to leave or it decides that it is the best to handle the client.

Node 5: This node is a member of the Data Group, and from its point of view, it is not the best node in the group. This node will send a Leave Request and continue to service the client until it receives an acknowledgment to leave or it decides that it is the best to handle the client.

Since nodes 4 and 5 are in disagreement from their own perspective, they will both service the client until one of them is able to take responsibility for handling the client. That is, none of these nodes can send acknowledgments to a Leave Request.

This mechanism guarantees that at least one node is a member of the Data Group, unless this node crashes. During disagreements, more than one node may be a member of the Data Group for some time, until the disagreement is resolved. Our experiments show that this usually lasts less than a quarter of a second during handoffs.

3.4 Experimental Results

3.4.1 Setup

We deployed SMesh on 15 Linksys WRT54G wireless access points across several floors in three buildings at The Johns Hopkins University. Only one of the routers was connected to the Internet. Each of the mesh nodes is equipped with one radio configured in ad-hoc mode. The data rate on the mesh nodes was set to legacy 11 Mbps 802.11b unless otherwise noted. The transmission power of the mesh nodes was set to 50 mW, and the 802.11 link-layer retransmission limit to 4. Unless specified, the topology of the mesh, depicted in Figure 3.4, was stable.

We used two laptop computers, each with a Broadcom 802.11g Mini-PCI card in ad-hoc mode as mobile clients. We used Linux for all experiments that required precise timing measurements. Windows XP was used for a TCP throughput experiment, also showing how SMesh operates across different platforms. No software other than the benchmarking programs was installed on the laptop computers.

The Linksys routers were modified with the available custom openwrt firmware [61]

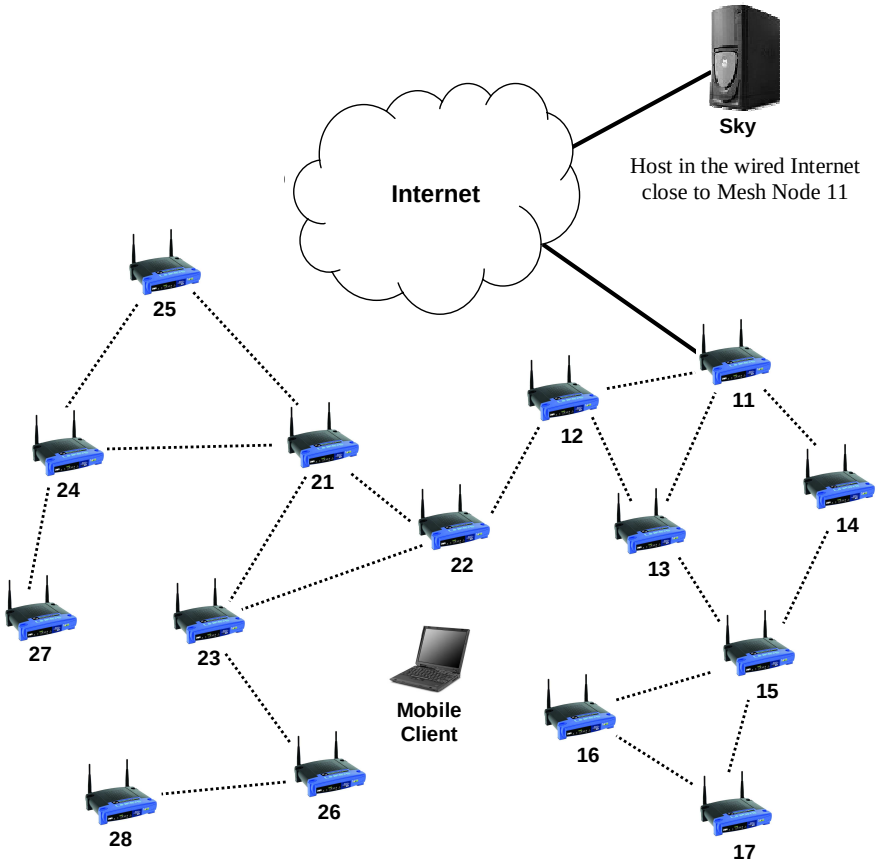


Figure 3.4: The SMesh Testbed.

that provided us with a Linux environment suitable for running the SMesh software. Other than adding SMesh, no other changes were made to the openwrt firmware.

The DHCP Server was set to issue lease times to clients for 90 seconds. The SMesh monitor was set to unicast ARP requests to the client and to use loss rate and RSSI in the client metric. For the link quality measure we used a $Current$ value of 50, and we set the decaying factor, D_f, to 0.80. The Threshold for joining the Client Data Group was set to 12%. In our experiments these numbers provided the best trade-off between the granularity of the metric and handoff responsiveness.

Our experiments were performed with one mobile client inside SMesh communicating

with a Linux machine that resided in the wired network (Internet), one wired hop away from the mesh Internet gateway. The SMesh client will be referred to as *Client* and the Linux box from the Internet as *Sky*. In the experiments we sent full-duplex VoIP traffic, one stream from Client to Sky and another from Sky to Client. The VoIP traffic consisted of 160 byte UDP packets sent every $20ms$ at a rate of $64Kbps$. This traffic is equivalent to that of G.711, the standard encoder used for VoIP communication.

We first performed a stationary test to set the baseline of our moving experiments. We then proceeded to move across two buildings starting and ending at the same location as the stationary experiment. We then show how TCP behaves as we move across the mesh. We tested the fail-over performance of our protocol when the access point of the *Client* suddenly crashes (we disconnected the power of the Linksys router). Finally, we added more mobile clients into the system, and determined how the management overhead of the mesh network increases as the system needs to handle more clients.

For each test we monitored the one-way latency of each packet, the number of lost packets, and the number of duplicate packets. The one-way latency was adjusted taking into account the difference between the clocks at the *Client* and *Sky* machines. For VoIP communication it was also important to track the delay jitter as well as how many packets arrived within $100ms$, the rest being considered lost by the audio codec. Based on tcpdump logs we reconstructed the handoff decisions and computed the communication overhead. We show the handoff information in the graphs, noting also the number of wireless hops from each mesh node to the Internet gateway. Note that the Client is connected to the access point through a wireless link, and therefore its latency is influenced by this additional link. When we state the number of hops of an access point we do not count the wireless hop from the client to its current access point.

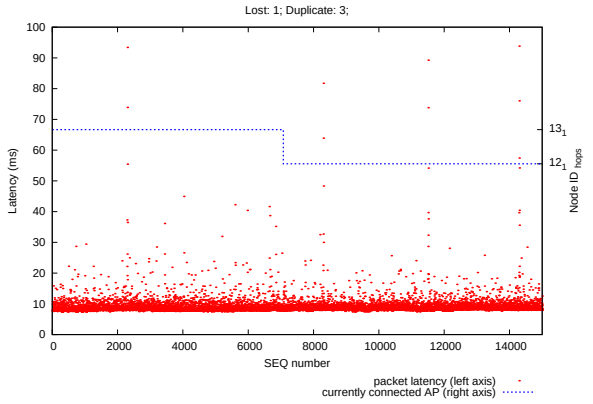


Figure 3.5: Stationary client. Mobile Client is the receiver.

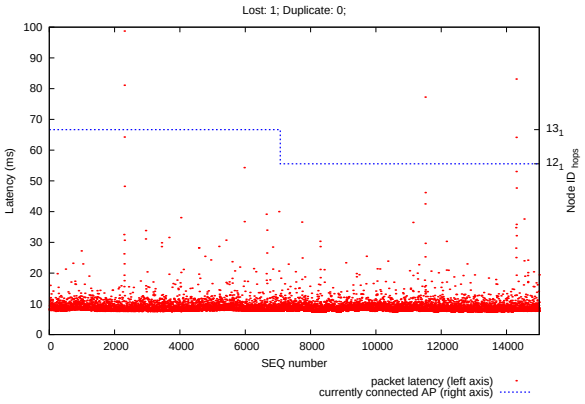


Figure 3.6: Stationary client. Sky is the receiver.

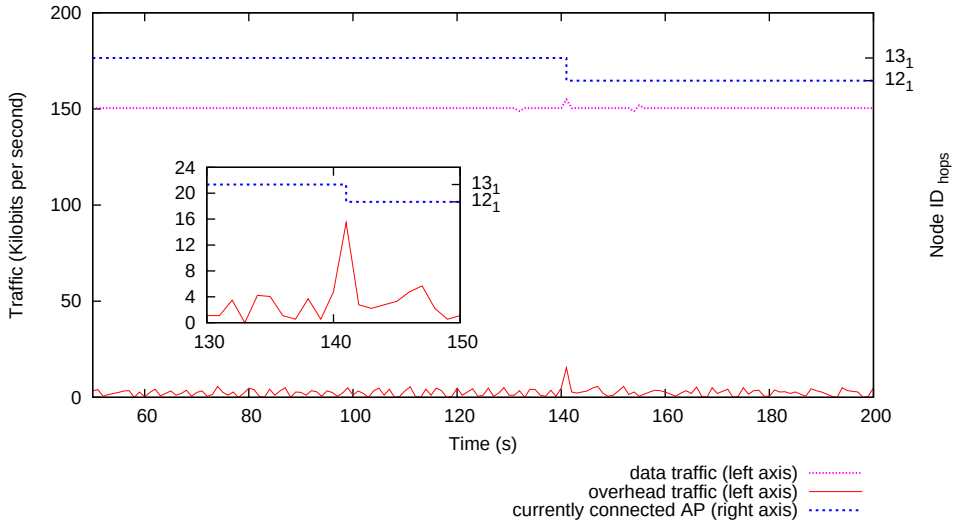


Figure 3.7: Stationary client. Data and SMesh Overhead Traffic. Subgraph shows traffic during handoff.

3.4.2 Measurements

Stationary client: This test was performed with the mobile client being stationary, in a fixed position for the duration of the entire test. UDP traffic consisting of 15,000 packets was sent simultaneously in each direction: from the Internet box (*Sky*) to the *Client*, and from the *Client* towards *Sky*. The packet latencies are shown in Figure 3.5 and Figure 3.6. The dotted line tracks which mesh node is the current access point of the *Client*. Vertical lines represent the moments when a gratuitous ARP that caused a handoff was sent. For example 12_1 on the right side of the graph refers to node 12 in our topology, which is 1 hop away from the Internet gateway. We notice that even though the client was stationary, its access point changed between two nodes in its vicinity: box 12_1, and then 13_1. This happens because the wireless connectivity varies, and over time, different access points have a better connection to the *Client*.

For the first stream (*Client* is the receiver, Figure 3.5), the number of lost packets was 1, and the number of duplicate packets was 3. This amounts to an overhead due to duplicates during handoffs of $.01\%$. During this experiment, 4 packets (0.02% of the total traffic) were delayed by more than $100ms$, and all packets arrived in less than $200ms$. As expected, the duplicate traffic occurred only during the handoffs[3].

The reverse stream (Sky is the receiver, Figure 3.6) had also 1 loss, but no duplicate packets. Only 1 packet arrived later than $100ms$, but before $200ms$. In all the tests when the Internet box (*Sky*) is the receiver, the number of duplicate packets must be zero: the packets are sent only once by the client (only to its current access point), in contrast to the other direction (from Sky to the Client) .

[3]We refer as "handoff" to the entire interval when duplicate packets are received; the time it takes the client to switch from one access point to another is as low as the time it takes for a gratuitous ARP to arrive from the access point to the client.

Figure 3.7 shows the overhead of our system in comparison with the data traffic. The data traffic represents the data traffic sent and received by the client during the experiment. The overhead traffic represents the data traffic sent, received, and forwarded by one of the mesh nodes in the client vicinity (mesh node 13). The bandwidth measured is higher than the full duplex 64Kbps UDP stream we sent, due to the IP and UDP headers that accumulate on the relatively small (160 byte) packets. (160 bytes per packet plus 8 bytes for the UDP header plus 20 bytes for the IP header gives us 188 bytes -1504 bits- per packet. With 50 packets per second each way, there are 9400 bytes -75200 bits- per second in each direction, or 18800 bytes -150400 bits- per second total).

Control traffic from our system is represented as the bottom traffic line. It combines the traffic from Spines (joins and leaves from multicast groups, hello keep-alive messages, link state updates) and the traffic from client's Control Group (link quality updates). Spines sends keep-alive messages of 40 bytes every 4 seconds. Link state updates are sent only when the mesh topology (formed by access points) changes. Join and leave messages are sent only when a SMesh node (access point) joins or leaves a group. These types of messages are aggregated such that a single Ethernet packet can contain up to 90 updates. In order to keep track of the clients (posting link quality measures, sending ARP packets), a SMesh node sends about 30 bytes per second (116 bytes in each update, sent every few seconds) for each client in its vicinity.

As we can see in Figure 3.7, a handoff takes place around second 140. The overhead during handoff is shown in detail in the zoomed graph on the left of the figure. The increase in control traffic show the moment when node 12 decided to join the Data Group, and sent a join message to Spines (join and leave operations will generate a state update in the Spines overlay network). As a consequence, there is a small spike in the data traffic since

data packets are duplicated. Right after, the old access point decided to leave the client Data Group (it sends a Leave Request and it immediately receives the acknowledgment). All of this happens in less than a second, so all of the overhead related to the handoff is represented by the spike in the control traffic during handoff.

We use the above stationary client results as a baseline for the following tests, to provide an idea of our wireless environment, and to overview the handoff process before a more elaborate scenario.

Moving client: In this test we move the client from the stationary position of the previous experiment, taking it on a 5 minutes trip across two floors and ending in the original position. We used the stairs to move between the floors. During the test, the client changed its access point 10 times, spanning from zero-hops away (11_0) to four-hops away (26_4). Note that the wireless hop between the client and its current access point is not counted in the number of hops in the network (so there is effectively one more wireless hop end-to-end).

The latency graphs for each of the two VoIP streams are shown in Figures 3.8 and 3.9 respectively. Each additional hop on the path from the Client to the Internet gateway resulted in an increase in packet latency: between sequence numbers 0 and 1315 we were zero hops away, between 1315 and 4298 one hop away, and between 4292 and 5794 two hops away. The number of packets that did not arrive within $100ms$ on the Client and Sky was 25 and 13, respectively. All packets arrived within $200ms$.

The data stream towards the *Client* had 3 packets lost, and 23 duplicate packets. Figure 3.10 presents the cumulative number of lost packets in a window of last 20 packets. The first loss occurred at packet 1419, about 100 packets after the handoff; this loss happened due due to loss in the medium. The second loss is far from any handoff, and happened due

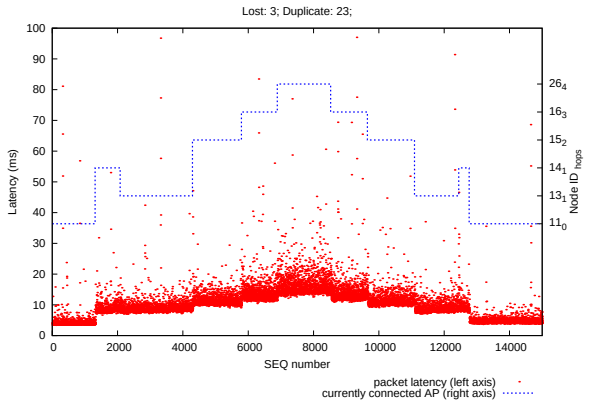


Figure 3.8: Latency. Moving client. Mobile Client is the receiver.

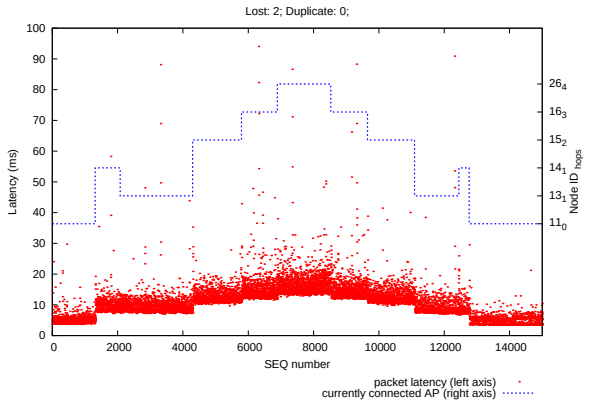


Figure 3.9: Latency. Moving client. Sky is the receiver.

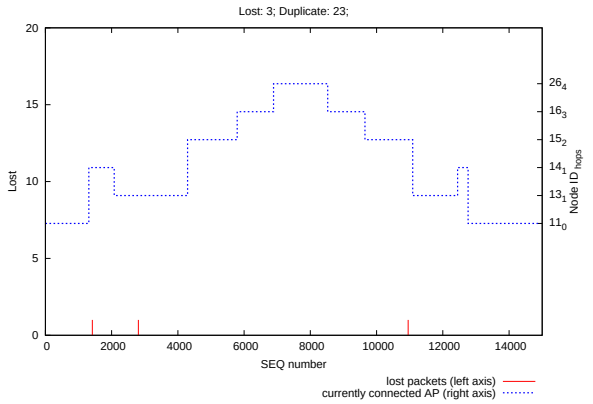


Figure 3.10: Lost packets. Moving client. Client is the receiver.

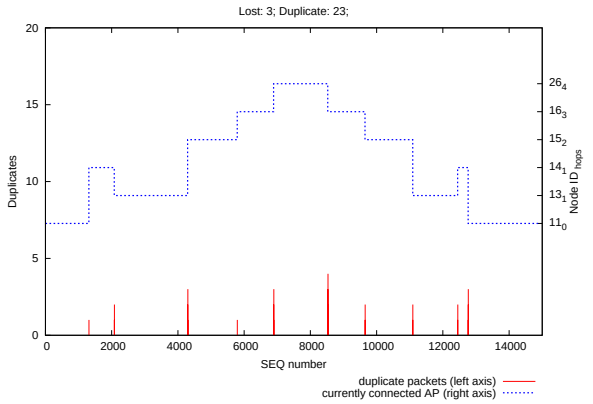


Figure 3.11: Duplicate packets. Moving client. Mobile Client is the receiver.

to loss in the medium as well. The third loss, however, happened at packet 10952, and the handoff started at packet 11094, or about 2.8 seconds after the loss. This loss contributed to lowering the metric, and to triggering the handoff. There is a possibility that the loss could have been prevented if the handoff would have happened earlier. While possible, our threshold (set at 12%) attempts to balance stability with handoff performance, and new information about a sudden drop in signal quality from a node takes time to propagate to other nodes. However, most of the handoffs were performed in a timely fashion without any loss before, during, or after the handoff. None of the losses in the experiment happened

during handoff itself.

Figure 3.11 shows the cumulative number of duplicate packets received in a window of last 20 packets. Note that duplicate packets happen only during handoffs. We can see that there is a correlation between the distance between the mesh nodes involved in the handoff and the number of duplicate packets. For example, nodes 15 and 16 are direct neighbors, and one to two duplicates were seen during a handoff between these nodes. In contrast, nodes 16 and 26 are further from each other, 6 wireless hops total, and three to four duplicates were recorded. In our approach, a node needs to learn about someone taking over the connection, request to leave, and receive an acknowledgement, before it can leave the data group associated with the Client. In addition, the multicast leave operation needs to propagate through the network.

The number of duplicates in our experiments show a lowerbound for our network; one can allow for more time to elapse before acknowledging a leave request to ensure that the state is fully propagated through the network before a multicast leave operation is issued. This will usually be a function of the diameter of the network and the timeouts for propagating state updates in each hop. While we did not experience any loss during handoff, allowing for longer period of time may be useful in other deployments.

The stream towards *Sky*, depicted in Figure 3.9, had 2 lost packets and 0 duplicates.

Figure 3.12 represents a zoomed view of the handoff happening at sequence 8526, for the same experiment. The dots represent the packets forwarded by the previous access point (node 26), and the crosses represent the packets forwarded by the new access point (node 16). The vertical line shows when the client received a gratuitous ARP from the new access point (node 16). This is the handoff that experienced the most number of duplicates, and between the nodes that are the most number of hops away from the Internet gateway

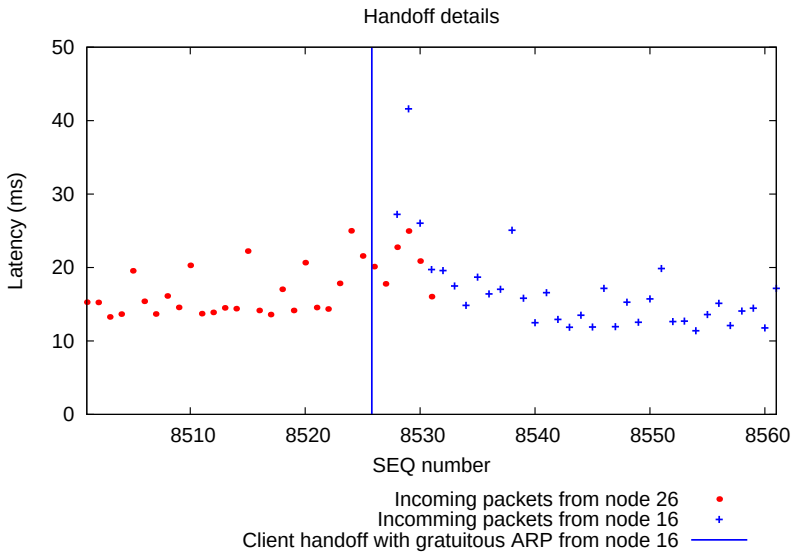


Figure 3.12: Zoom during handoff. Moving client. Mobile Client is the receiver.

and from each other.

The gratuitous ARP from node 16 was received just before packet 8526. However, there are no duplicates until packet 8528. When a node makes a local decision to start handling the client, it issues a gratuitous ARP in addition to a multicast join for the data group for the client. However, this multicast join needs to propagate and routes need to be established before packets start flowing towards the new access point. It took between 20ms and 40ms for this to happen, which is consistent with the number of hops between the nodes and our choice of timers[4] in the system. When communicating with a node in the Internet, this delay depends on the number of hops from the node joining the Client data group and the Internet gateway. Starting at packet 8528, there are four duplicate packets received by the Client. As previously explained, the number of duplicate packets depend on the number of hops between the nodes involved in the handoff. We can see a slight increase in latency

[4]In Spines, each overlay node waits 5ms before forwarding the update to it's neighbors. This allows the overlay to aggregate updates and scale.

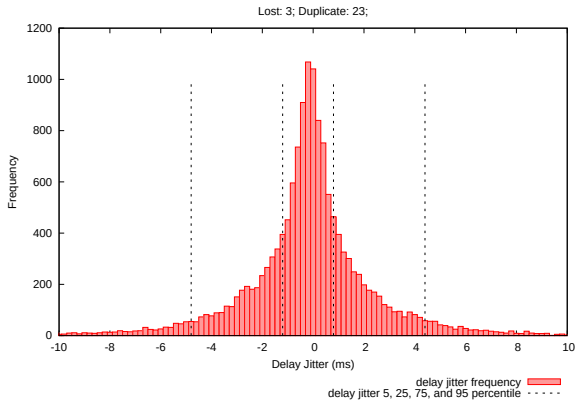


Figure 3.13: Delay Jitter. Moving client. Client is the receiver.

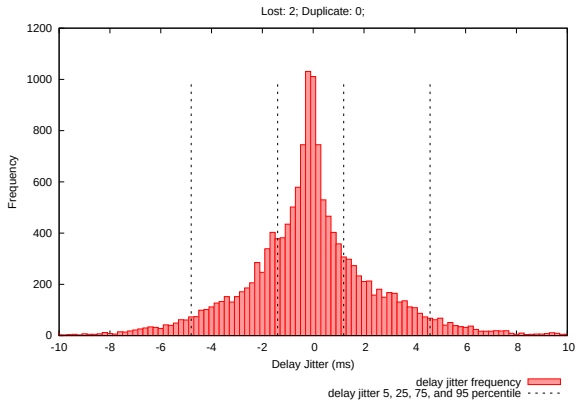


Figure 3.14: Delay Jitter. Moving client. Mobile Client is the receiver.

during handoff, which is caused by the increase in congestion in the wireless network. The latency drops slightly afterwards as the new access point is one hop closer to the Internet gateway.

Figure 3.13 and 3.14 show the delay jitter, or Inter Packet Delay Variation [62], of the VoIP stream towards the $client$ and towards sky, respectively. A big variation can have a negative effect on the playout buffer at the end-points of the VoIP stream. In our test, the Inter Quartile Range (IQR), which represents the difference between the 25 and the 75 percentile, was just 2.6ms. One can also see in Figure 3.12 that there is a light increase in jitter during handoff. Considering that a jitter of less than 20ms is considered excellent by VoIP applications, the quality of the voice is not impacted by the jitter experienced in the mesh network.

TCP handoff: In the next experiment, we used an 802.11g wireless card in the mobile client, and configured the mesh to 802.11g with a fixed rate of 36Mbps. We moved the *Client* throughout two floors, but this time going down and then up through different stairs in opposite sides of the building.

Figure 3.15 shows the TCP download throughput experienced by the mobile client.

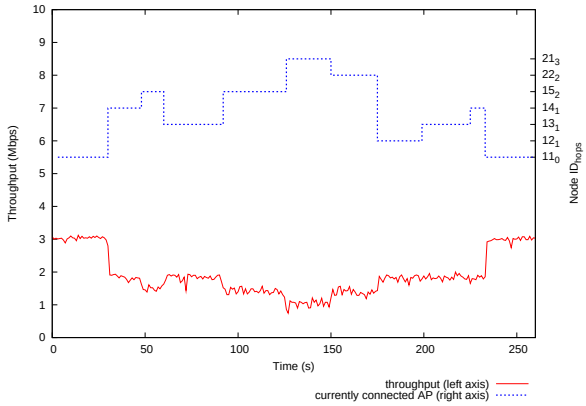


Figure 3.15: TCP throughput. Moving client. Mobile Client is the receiver.

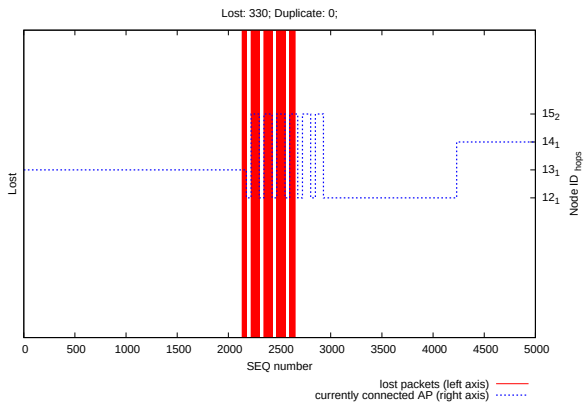


Figure 3.16: Mesh node and topology failover. Lost packets. Sky is the receiver.

Note that we move through the vicinity of a different set of nodes in this experiment. There were also 10 handoffs during this experiment. The throughput is just 3Mpbs when the client is connected directly to the Internet gateway, which is lower than expected. This bandwidth is a CPU limited amount; the CPU is 100% utilized at this point. As the number of hops increases, the throughput goes down to about 1Mbps. The throughput returned back to the original amount when we reached the original location where we started the test. TCP connection remained open at all times, and packets kept flowing regularly.

Fail-over: In this experiment we evaluated the fail-over performance of our system when the access point currently serving the client suddenly crashes. We used a stationary client connected to access point 13, sending a VoIP stream to the *Sky* box. As the *Client* was sending packets, we suddenly disconnected the power at node 13. Figure 3.16 shows the packets lost at *Sky* from the *Client* when node 13 fails. We can see that there are 5 intervals of loss close to each other. The first loss interval occurs as the *Client* keeps sending packets to node 13 after it fails. Shortly thereafter, node 12 notices it does not receive link quality measures from node 13, and sends a gratuitous ARP to the client, forcing its handoff. In our topology, the minimum hop distance routing selects the route between nodes 12 and 15

to go through nodes 13. After node 13 crashed, node 12 and node 15 do not receive link quality updates from each other, until routing in Spines is repaired. Therefore, both nodes 12 and 15 believe they have the best link quality to the client. They both insist on taking over the connection from the client, sending ARP messages to it. We can see this behavior in the six handoff oscillations depicted in Figure 3.16. Since node 15 does not have a route to the Internet gateway until the routing protocol in Spines detects the failure (its original route went through node 13), whenever it takes over the *Client*, the data packets are lost. This explains the following intervals of loss after the initial handoff. After Spines detects the failure and the network routes are fixed, packets from the *Client* are no longer lost (both nodes 12 and 15 can reach the Internet gateway). However, it takes a few more seconds for nodes 12 and 15 to send their link quality measures to each other and decide which one should serve the *Client*. Indeed, Figure 3.16 shows three more handoffs between nodes 12 and 15 until 12 is selected to serve the client.

Overhead: In this experiment we measured the management overhead of the system as additional mobile clients are introduced into the network. We evaluated the control traffic required to propagate routing and group membership information, to handle client mobility, and to maintain network's topology. We focus on the overhead of the *control* traffic, as the overhead caused by duplication of data packets was discussed in the previous experiments.

There are five main components of the control traffic:

- Hello messages: The mesh nodes send beacon messages of 40 bytes every 5 seconds in order to discover changes in the topology (node crashes or additional nodes in the system). This traffic does not depend on the number of mobile clients in the system, nor on their mobility.

- Link state updates: These messages propagate information about topology changes.

The state update messages are small (under 36 bytes), and multiple states are aggregated in a single packet whenever possible. Since the mesh nodes are stationary and the topology is relatively stable, and because we use reliable state updates, this overhead is negligible (basically 0 in our experiments). Therefore, we do not consider it in our analysis.

- Group state updates: These are the messages used to exchange group membership information between the nodes. The state update messages are also small (under 36 bytes), and multiple states are aggregated in a single packet whenever possible. The number of group state updates is highly related to the mobility and the number of clients. As a client moves, some mesh nodes will join its *Control Group* and *Data Group*, while others will leave.

- Gratuitous ARP messages: Gratuitous ARP messages are sent by the members of *Data Groups* as described in Section 3.3.3. The size of an ARP packet is 28 bytes. As mobile clients change their access points as they move, the ARP traffic depends mostly on the number of the clients and their mobility.

- Monitoring messages: These are ARP heartbeat packets that are sent (and received) by the access points to assess the quality of the link with a client. In our experiments an access point probes a client every second. This component of the control traffic increases linearly with the number of clients.

- Link Quality updates: Nodes in the vicinity of a mobile client send 68 byte messages periodically, to share information about the link quality between the members of the *Control Group*, and during handoff. The Link Quality traffic depends on the number of clients and their mobility.

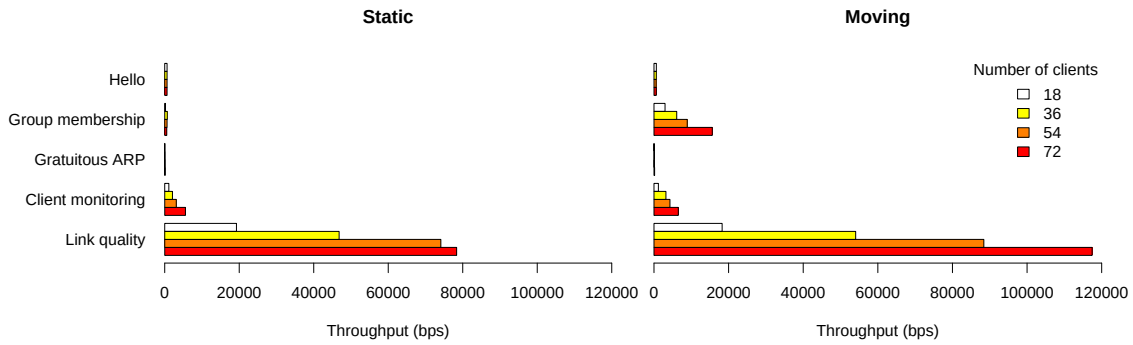


Figure 3.17: Overhead traffic.

We first measured the overhead traffic in tests with different numbers of stationary clients: 18, 36, 54 and 72. The clients were evenly spread in the mesh network, which corresponds to 1, 2, 3 and 4 clients, respectively, connected to 18 access point in our testbed. To support such experiments, which require a large number of simultaneous users, we implemented a client emulator that generates the appropriate control traffic associated with a regular client. From the 802.11 network and from the system's perspective, there was no difference between an emulated client and a real client in terms of control traffic. In the second test we evaluated the system while the clients were moving through the coverage area, each one randomly switching its access point about every minute.

Our measurement reflect the traffic seen by a single mesh node, node 11. For each type of traffic, we measured the overhead traffic considering the full size of the packets including the IP and UDP headers. Figure 3.17 illustrates the overhead traffic as the number of clients increases for static and moving clients. Table 3.1 shows the average number of packets per second sent and received for each type of overhead traffic, and the corresponding average throughput rates are shown in Table 3.2.

In the stationary tests, the highest bandwidth consumer was the link quality update traffic. The average throughput of Link Quality messages per second increased from

	Hello	Joins/ Leaves	Gratuitous ARP	Link monitoring	Link Quality	**Overall**
18 clients, stationary	1.16	0.36	0.15	4.93	25.11	**31.72**
36 clients, stationary	1.16	1.34	0.29	9.26	60.97	**73.02**
54 clients, stationary	1.16	1.23	0.41	13.85	96.65	**113.30**
72 clients, stationary	1.17	1.06	0.52	24.84	102.18	**129.77**
18 clients, moving	1.16	5.72	0.11	5.26	23.65	**35.90**
36 clients, moving	1.15	11.71	0.39	14.29	70.32	**97.86**
54 clients, moving	1.06	17.09	0.24	19.11	15.25	**152.74**
72 clients, moving	1.14	29.52	0.75	29.12	153.15	**213.68**

Table 3.1: Average number of packets sent and received per second for each type of overhead traffic.

19,258 bps to 78,362 bps. The second worst consumer (although five times less) was the link monitoring traffic, which linearly increased from 1,106 bps to 5,564 bps. The rest of the traffic is low: as expected, the *hello* protocol has a constant overhead, which amounted to approximative 633 bps (1.16 average messages per second) while the traffic generated by joins/leaves stayed below 700 bps. The gratuitous ARP traffic was almost zero as the clients were stationary. Overall, the average overhead increased linearly with the number of clients, from 20.7 kbps for 18 clients to 82.2 kbps for 72 clients.

In the tests with moving clients, the highest bandwidth consumer continues to be link quality traffic – with a maximum of 117,436 bps for 72 clients. However the second is now the group membership traffic, which grows from 2,962 bps for 18 clients to 15,627 bps for 72 clients. This is because the movement of the clients resulted in increased activity on their *Client* and *Data Groups*. Since the clients moved randomly in the network, the density per node stays about the same throughout the experiments, therefore the link quality monitoring

	Hello	Joins/ Leaves	Gratuitous ARP	Link monitoring	Link Quality	**Overall**
18 clients, stationary	629.74	183.65	34.09	1,105.39	19,257.22	**21,210.09**
36 clients, stationary	633.42	690.70	64.44	2,074.30	46,818.48	**50,281.35**
54 clients, stationary	632.56	634.23	91.16	3,102.14	74,124.84	**78,584.93**
72 clients, stationary	635.08	551.53	117.57	5,563.51	78,361.48	**85,229.18**
18 clients, moving	632.50	2,961.15	23.58	1,177.88	18,264.80	**23,059.90**
36 clients, moving	625.60	6,098.08	86.24	3,202.08	54,061.44	**64,073.44**
54 clients, moving	578.97	8,912.00	52.80	4,280.00	88,414.86	**102,238.63**
72 clients, moving	617.70	15,626.32	169.08	6,523.46	117,436.49	**140,373.06**

Table 3.2: Average throughput rates for each type of overhead traffic. Results are in bps.

traffic is about the same as in the stationary test. The gratuitous ARP traffic is higher than before (each client experienced a handoff approximately every minute, which corresponds to more than one handoff per second in the entire network) but overall is extremely low. The network topology remained unchanged causing the same amount of hello traffic. Overall, the average overhead increased linearly with the number of clients, from 22.5 kbps for 18 clients to 127.1 kbps for 72 clients.

The aggregate management overhead increases linearly with the addition of clients, from 1.4 kbps per client for stationary clients, to 1.9 kbps per client for moving clients.

Finally, we demonstrate the operation of the system in a more diverse scenario, and show the overhead traffic sent and received by node 11 during the experiment (Figure 3.18). We started with no clients in the system (section A), and then gradually added 72 stationary clients evenly spread in the mesh network (section B). All the clients are stationary for some time (section C), after which half of them started to move (section D).

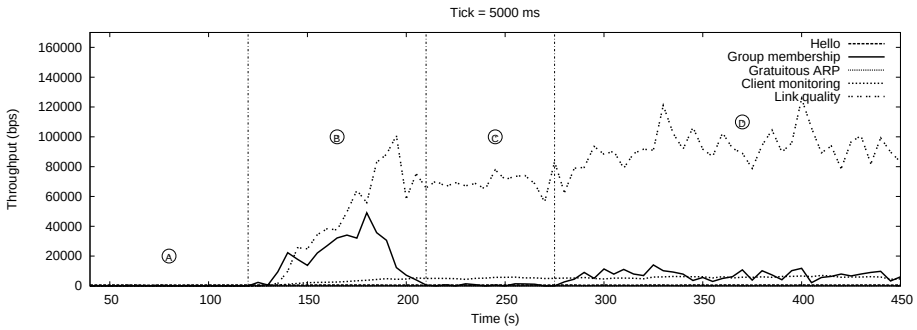


Figure 3.18: 18 nodes, 72 clients. Overhead Traffic. (A) no clients, (B) 72 clients connect, (C) all clients are stationary, (D) 36 of the clients start moving throughout the mesh.

Throughout the experiment, the hello and gratuitous ARP traffic stays very low, compared to other components. As clients join the network, we see a small increase in the client monitoring traffic, which remains stable after all the clients are connected. In contrast, as clients join the network, we see a significant increase in group state update traffic due to mesh nodes joining the *Control* and *Data* groups for the clients in their vicinity. Because the clients are stationary, this traffic goes back to zero after the updates are propagated in the network (section C). However, when some of the clients start to move (section D), the group state overhead traffic increases again as an effect of membership changes in the *Control* group (due to new clients coming within the vicinity of mesh node 11) and the *Data* group (due to handoffs). In the same way, link quality traffic increases while the clients join the network, but afterwards remains high since mesh nodes periodically share link quality information. We notice a small increase in this traffic when clients start to move (section D), mainly due to more clients coming within the vicinity of node 11.

Experiments summary: The experiments show that the SMesh protocols provide instantaneous handoff, with a low overhead caused by duplicates during periods of instability caused by handoffs. When sending and receiving both UDP and TCP traffic, the connections were not interrupted, and the loss when a mobile client roams was minimal.

As expected, a short disconnection happens when the access point serving the client suddenly crashes. In such a case, the system re-adjusts, and within a few seconds is able to re-route packets through the network.

The management overhead of the mesh network grows linearly with the number of clients, in the worst case at a rate of about 2 kbps per client. This overhead does not depend on the amount of data the mobile clients send or receive. Considering that the capacity of 802.11g wireless networks is in the order of tens of Mbps, we conclude that the management overhead of SMesh is reasonable.

Chapter 4

Achieving Fast Inter-domain Handoff

This chapter present the protocols that we developed to support hybrid routing and fast inter-domain handoff in multi-homed wireless mesh networks. The protocol integrates wired and wireless communication and optimizes performance of the hybrid routing, in our case by minimizing the usage of wireless transmissions.

We start by overviewing multi-homed wireless mesh networks, and describe our hybrid overlay architecture, including topology formation and hybrid routing metric. We then describe our inter-domain handoff protocol, and how TCP and UDP connections need to be treated differently to maintain connectivity. Finally, we demonstrate that inter-domain handoffs occur instantaneously, with virtually no loss or delay, for both TCP and UDP connections.

4.1 Multi-homed Wireless Mesh Networks

A wireless mesh network extends the connectivity range of mobile devices by using multiple access points to create a mesh topology and forward packets over multiple wireless

hops. As the size of a wireless mesh network increases, the number of Internet connected access points (*Internet gateways*) needs to increase to disperse traffic and avoid congestion. In practice, Internet gateways will reside at different locations and will often be connected to different network domains. We refer to such mesh networks as *multi-homed*. In this type of networks, a mobile client is served by a nearby access point that forwards data packets (potentially over multiple wireless hops) to its closest Internet gateway.

Multi-homing poses a challenge in providing continuous connectivity to mobile clients that may move between the areas covered by different access points. Those access points will often have different Internet gateways closest to them. When such a transition (*hand-off*) occurs, we would like to maintain all previously opened connections, and transfer them to the new Internet gateway as quickly as possible, without any involvement from the mobile device.

In our approach, new connections always use the closest Internet gateway at the time of their creation, while existing connections are forwarded through the wired infrastructure to the Internet gateway where they were originally initiated. As the handoff process requires routing agreement and transferring connections between the involved Internet gateways, our protocol guarantees that packets are routed correctly, at all times.

4.2 A Hybrid Overlay Architecture

A wireless mesh network is comprised of multiple access points, possibly distributed in several islands of wireless connectivity such as different buildings located close to each other or parts of the same building. Access points inside a wireless island can communicate, potentially using multiple intermediate hops. One or more access points in each wireless

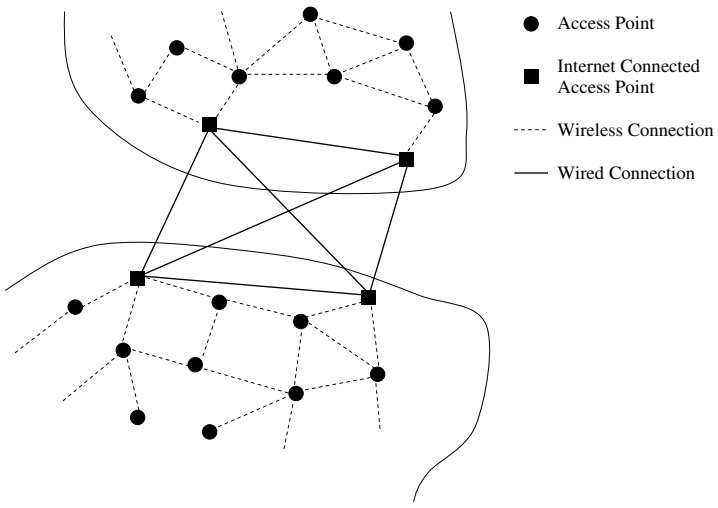


Figure 4.1: Hybrid Overlay Mesh Network

island is connected to the Internet through a wired network. For Internet connectivity, other access points rely on multi-hop communication to reach an Internet Gateway in their island. Figure 4.1 shows an example of a wired-wireless hybrid mesh network with two islands, each of them with two Internet gateways.

Each access point runs a software router that allows multi-hop communication. These routers create an overlay topology where some of the links are wireless (between nodes in the same island) while others are wired (between the Internet gateways). In our implementation we use the Spines overlay messaging system to provide multi-hop communication as it offers overlay multicast, anycast and unicast forwarding. We make use of overlay multicast to auto-discover Internet gateways and to coordinate decisions between access points during mobile client handoffs. We use anycast to forward data packets from a client to the closest Internet gateway.

Using one overlay network for both wireless and wired communication has several advantages. Peer-to-peer communication between access points located in the same wireless island can take advantage of wired connectivity between remote Internet gateways to short-

cut multiple wireless hops. In addition, the diameter of the network is decreased, improving route update latency and overhead related to control messages on the overlay network.

4.2.1 Topology Formation

The topology formation starts with each access point broadcasting its presence periodically. Neighboring nodes create bidirectional links and advertise their connectivity through a link state protocol to other nodes in the network. The link state protocol uses link-based acknowledgments such that after a link was advertised to other access points in the network, it will not be advertised again, unless it changes its status. This reduces communication overhead for managing the topology.

Internet gateways join a multicast group called *Internet Gateway Multicast Group* (IGMG) on which they periodically advertise their wired interface IP address. The multicast routing is handled by the underlying overlay infrastructure, as explained in the previous chapter. When two Internet gateways receive each other's advertisements (which initially travels through the wireless infrastructure to the members of the multicast group), they connect through a wired overlay link. This way, the Internet gateways inside an island form a fully connected graph using their wired infrastructure, while the other access points inside the island interconnect based on the wireless connectivity. In order to interconnect wireless islands, at least one Internet gateway in each island needs to be pre-configured to connect to a set of Internet gateways such that an initial connected graph is formed. Then, multicast advertisements from all gateways will be propagated, Internet gateways will connect to each other, and eventually, a fully connected logical graph between all Internet gateways in all islands is formed.

4.2.2 Routing Metric

In a multi-homed wireless mesh network, some of the access points have wired connections that can be used to shortcut several hops of wireless communication, thus decreasing the number of wireless transmissions. In general, in a combined wired-wireless routing metric scheme, it is reasonable to assume that a wired connection costs much less than a wireless link. On the other hand, depending on the network conditions it is possible that wired connections between Internet gateways have different costs (based on throughput, loss rate, latency, etc.).

Our approach uses the best route to a destination considering wireless connectivity as well as any hybrid route available, and allows for different routing metrics to be used both on the wired and wireless links. Considering that each wireless link can have an $ActualCost$ metric of at least 1, the routing cost of that link will be:

$$Cost = ActualCost * (M + 1)$$

where M is the maximum cost that can be associated with a wired path. For example, if a wired link can have a maximum cost of 10, and there are 5 access points connected to the Internet in the mesh network, the value of M is 40 (the largest number of wired hops in a path is 4), and the minimum cost of a wireless link is 41. The cost of a hybrid path is the sum of the cost of all the links. This mechanism gives preference to any wired link over a wireless one, and optimizes the wired path based on a desired metric. For example, we can use ETX [63] as the wireless $ActualCost$ metric, and latency as the wired links metric.

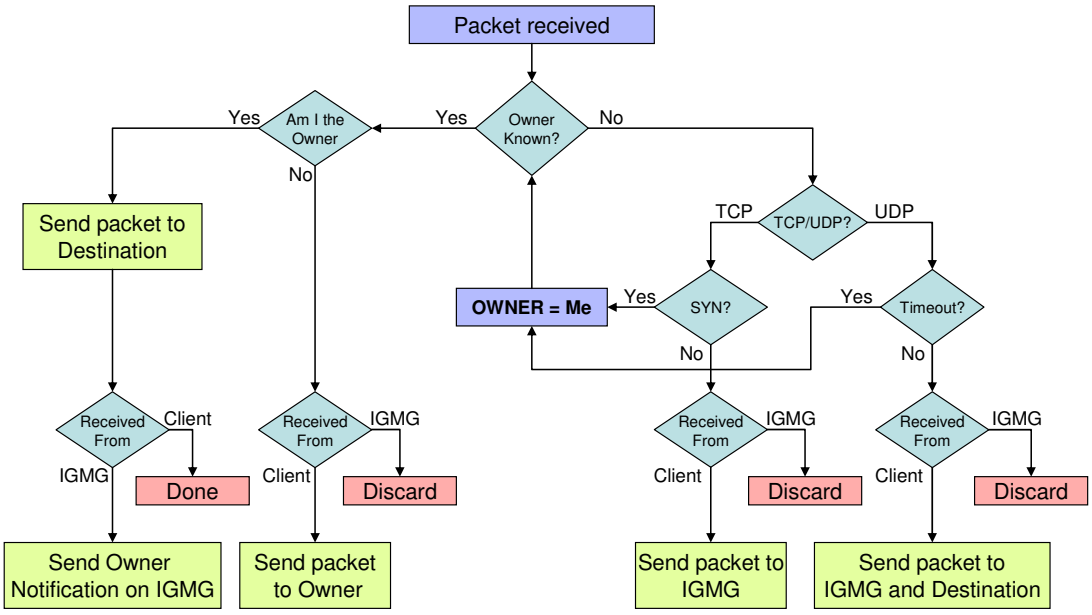


Figure 4.2: Inter-domain Handoff Flowchart

4.2.3 Handling Mobile Clients

As previously explained in Chapter 3, Mobile clients connect to their closest access point and use it transparently as they would work with a regular Internet connected access point. No special software or drivers need to be installed on the mobile clients. The mesh network is responsible to forward packets to and from other clients or the Internet. In our implementation, all access points use a private IP domain (10.x.y.z) for their wireless interfaces. Mobile clients are assigned IP addresses through DHCP from the same IP domain.

Packets sent to a mobile client are routed by the overlay infrastructure to the Data Group corresponding to the receiver client. Local access points that joined the Data Group then forward the packets to the mobile client. The reason for using a multicast group instead of a single IP address for the client packets is that in periods of instability, when it is not yet decided which local access point should serve the client, multiple access points in

the vicinity of the mobile client may forward the data packets (also allowing us to deal with unpredictable moving patterns). When an access point receives a packet that has a destination outside the wireless mesh network, it simply forwards it to the Internet Gateway Anycast Group, an overlay anycast group to which all Internet gateways join. This way, packets are always sent to the closest Internet gateway.

4.3 Inter-domain Handoff Management

4.3.1 Internet Gateway Control Group

Packets exchanged between two mobile clients, either in the same or in different wireless islands, simply use shortest path multicast trees reaching the access points that decided to serve each client. Note that in the stable case, when mobile client communication does not require a handoff, only one access point in the vicinity of a client will join its multicast Data Group. Therefore, most of the time, the multicast trees are simply linear paths. The multicast trees adjust automatically when mobile clients roam within the vicinity of different access points, as the access points join or leave the client's multicast Data Group. In peer-to-peer communication, packets will follow the shortest paths with no need for a special handoff at the Internet gateways.

In contrast, communication between mobile clients and the Internet is relayed through the closest Internet gateway. As mobile clients move within the wireless mesh network, they may get closer, network-wise, to a different Internet gateway in the same island, or they may move to a different wireless island. In this case, the anycast packets, which are forwarded to the closest Internet gateway, will no longer reach the original gateway, and therefore a solution is required to maintain existing connections.

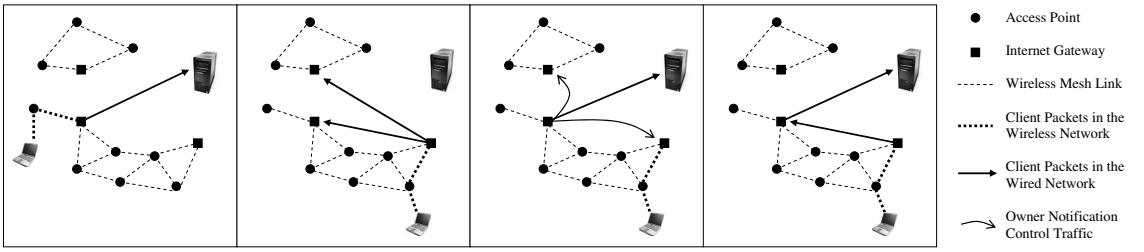


Figure 4.3: TCP forward handoff: (a) Connection establishment (b) Handoff Phase 1 (c) Handoff Phase 2 (d) Handoff completed

Mobile clients in SMesh work on a private network, and a Network Address Translation (NAT) is required at the Internet gateway when communicating with an external host. Each Internet gateway has a different external IP address. Applications using TCP, and in some cases, applications running on top of UDP require packets to be forwarded through the initial forwarding Internet gateway through the entire life of the connection. Changing one end-point of the connection (the IP address of the Internet gateway) is often impossible without breaking the existing connection, and therefore it is better for the handoff mechanisms to mask this problem inside the mesh network.

One potential solution is to exchange complete connection information (NAT tables) between the Internet gateways periodically and forward packets to the original owner of the connection using the wired connectivity. Such a solution can only be as fast as the time between two periodic NAT table exchanges, and cannot support real-time traffic such as VoIP. To support real-time traffic, one can advertise connection information to all the Internet gateways when the NAT entries are created. However, this technique tends to be wasteful, as not all mobile clients may move and change their Internet gateway. The problem is most notable when clients are browsing the Internet, as many connections are established for each website and, all of these information, which is relevant only for a small amount of time, would be sent to all of the Internet gateways.

Our inter-domain handoff protocol provides transparent mobility on a NATed network with real-time performance. We treat UDP and TCP connections separately, detect the existing owner (the Internet gateway from which the connection was initiated) of a connection, and forward existing connections through their original owners[1]. Figure 4.2 shows the general flow of packets at each Internet gateway.

4.3.2 TCP Connection Handoff

A TCP session requires that source and destination IP addresses and ports remain constant during the life of the connection. Our mobile clients run in a NAT address space, and although connections are end-to-end, the Internet destination regards the source address as that of the Internet gateway that sent the first SYN packet. When a mobile client moves closer to a different Internet gateway, the new gateway must forward all packets of each existing connection to the original gateway that initiated that connection. On the other hand, new connections should use the Internet gateway that is closer to the client at the current time, and not be forwarded to an old gateway.

In TCP, a SYN packet indicates the creation of a connection and generates a NAT entry, while a FIN packet indicates the destruction of the connection. If an Internet gateway receives a TCP packet that is not a SYN and it does not have an entry for that connection in its NAT table, it forwards that packet to the IGMG group. The original owner of the connection (the one that has it in its NAT table) relays the packet to the destination, and sends a message to the IGMG group, indicating that it is the connection owner for that NAT entry. Then, any gateway that is not the connection owner, will forward packets of

[1] One can potentially spoof the address of the original owner to reduce the routing overhead of our protocol. However, egress filtering is commonly used at network routers and will prevent spoofed packets from leaving their network.

that connection to the respective owner, finalizing the connection handoff process. Figure 4.3 shows the stages of such a TCP connection handoff.

If packets arrive at an Internet gateway at a fast rate, several packets may be sent to the IGMG group before the connection owner can respond. If no Internet gateway claims the connection within a certain timeout (in our implementation 3 seconds), the new gateway claims the connection, forwarding the packets directly to the Internet destination. This will break the TCP connection, which is the desired behavior in such a case, since it is likely that the original owner crashed or got disconnected. Causing the Internet host to close the connection avoids connection hanging for a long period of time (TCP default is 2 hours).

4.3.3 UDP Connection Handoff

Most real-time applications use the best effort UDP service and build their own protocol on top of UDP to meet specific packet latency requirements. Some applications, such as DNS, do not establish connections between participants. Others, such as SIP in VoIP, establish specific connections defined by a pair of an IP address and a port at both ends of the connection.

When an Internet gateway receives a UDP packet with a new pair of source and destination addresses or ports, it cannot distinguish between the case where this is the first packet of a new connection, and the case where the packet belongs to an existing connection established through a different Internet gateway.

We classify UDP traffic on a port number basis as *connection-less* and *connection-oriented*, and choose connection-oriented as the default protocol. Connection-less UDP traffic is forwarded directly after receiving it from the mesh network, on the current shortest path. DNS and NTP traffic falls into this category.

Upon receiving a new connection-oriented UDP packet that has an Internet destination, an Internet gateway relays that packet to its destination, and also forwards it to the multicast group that all Internet gateways join (as opposed to the TCP case, where the access point only sends packets to the multicast group). If the UDP packet belongs to a connection that was already established, the Internet gateway that is the original owner of the connection also relays the packet to the destination, and sends a response to the Internet gateway multicast group. After receiving the response, the initial gateway will forward subsequent packets directly to the original gateway, and will no longer relay UDP packets of that connection (with the same source and destination addresses and ports) to the Internet. If a response does not arrive within a certain timeout (in our implementation 500 milliseconds), the Internet gateway will claim ownership of the UDP connection, will stop forwarding packets of that connection to the IGMG group, and will continue to relay packets to the Internet.

4.3.4 Overhead

Internet gateways generate some overhead traffic on the wired network during the inter-domain handoff. Data packets are multicasted over the wired network to all other Internet gateways until the owner of the connection responds. In our tests, this process took between 10 ms and 60 ms. Note that data packets are forwarded in parallel to the end-host and their latency is much less. After the first handoff of a connection takes place, all Internet gateways are informed about the owner of that connection, and therefore new data packets are sent directly to the connection owner. As opposed to the wireless intra-domain overhead, which is only dependent on the number of clients, the inter-domain overhead is directly proportional to the number of connections each client has. However, the traffic generated

by the inter-domain handoff is small, and uses only the wired network.

4.3.5 Discussion

Due to handoff and/or metric fluctuations, there is a possibility that packets coming from a mobile client and belonging to the same flow alternate between two Internet gateways. This may lead to more than one gateways claiming the ownership of the connection. We encounter such case in TCP when a client retransmits a SYN connection request, and this request is routed through a different Internet gateway. In UDP, such case may occur when two different Internet gateways start forwarding client packets for the same connection at about the same time. A plausible solution for TCP is to delay ownership decision until a full three-way TCP handshake is seen by the Internet gateway. For UDP, when there is more than one ownership request in parallel, the gateways decide the rightful owner of the connection based on feedback traffic from the end-host or lowest IP address.

Also note that, in general, our inter-domain handoff protocol can be applied in less sophisticated architectures. For example, all Internet gateways can be pre-configured with the complete set of Internet gateways that will participate in the inter-domain handoff. However, route optimizations provided by the overlay network, both in the wired and wireless network, will not be available, and some other mechanism must be devised to ensure fast seamless handoff for mobile clients at the intra-domain level.

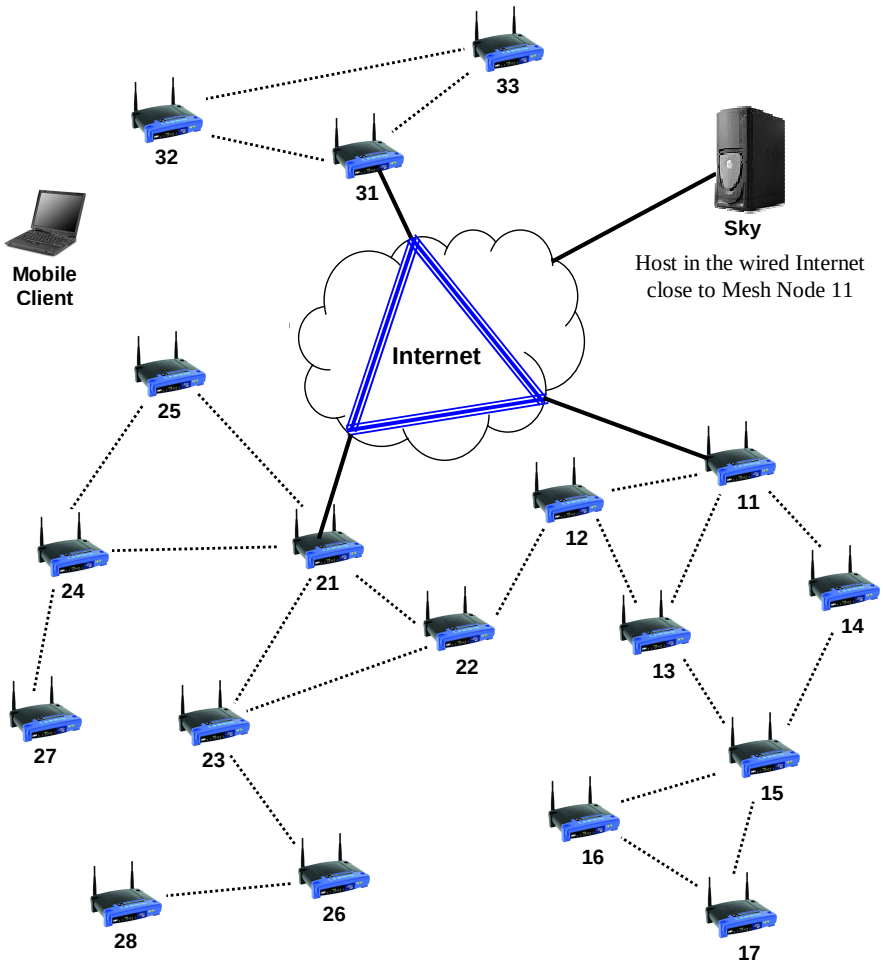


Figure 4.4: The SMesh Multi-homed Wireless Mesh Testbed.

4.4 Experimental Results

4.4.1 Setup

We deployed our system on 18 Linksys WRT54G wireless routers across several floors in four buildings. Each of the routers is equipped with one radio configured in ad-hoc mode. Transmit power of the access points was set to $50mW$. The Linksys routers were modified with the available custom openwrt firmware [61] that provided us with a Linux

environment suitable for running the SMesh software. Other than adding SMesh, no other changes were made to the openwrt firmware.

We used two laptop computers, each with a Broadcom 802.11g Mini-PCI card in ad-hoc mode as mobile clients. We used Linux for all experiments that required precise timing measurements. Windows XP was used for a TCP throughput experiment, also showing how SMesh operates across different platforms. No software other than the benchmarking programs was installed on the laptop computers.

The topology of the wireless testbed used in our experiments is shown in Figure 4.4. The topology consists of one main island with two Internet gateways, and another smaller island with one Internet gateway. The islands are disconnected due to a large open grass area between the buildings. However, a mobile client located between the two islands can reach both networks. Each of the Internet gateways is part of a different domain on the campus network and within 6 hops of each other through the wired network. Unless otherwise specified, the topology between the access points was static during the experiments. Each access point box has an identifier, refered to as node id. The node-id of Internet gateways ends with digit 1 (mesh nodes 11, 21, and 31). The closest Internet gateway of mesh nodes is given by the prefix of the access point box-id (i.e. node 23 uses node 21 as its Internet gateway). In addition, the node ids are ordered by number of hops from the gateway (i.e., node 23 is equal or less number of hops from from its gateway than node 24).

Experiments consist of walking with a mobile client from the 3^{rd} floor of a building located in the main island to a hallway in the second floor, followed by going down to the ground floor. Then, while walking outside on an open grass area we end up reaching the second island. This movement results in a few access point handoffs and at least three Internet gateway handoffs. A mobile client will be referred to as *Client* and the Linux box

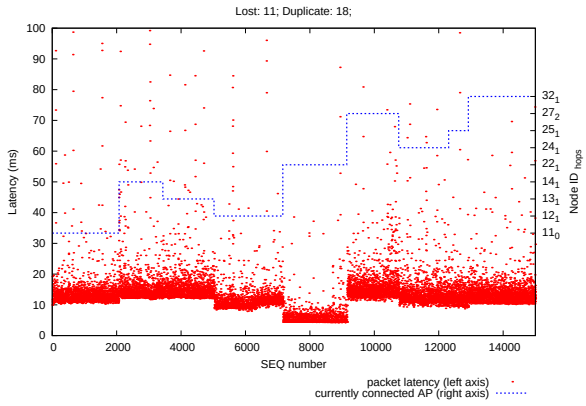


Figure 4.5: P2P Test. Latency of packets received at Moving Client.

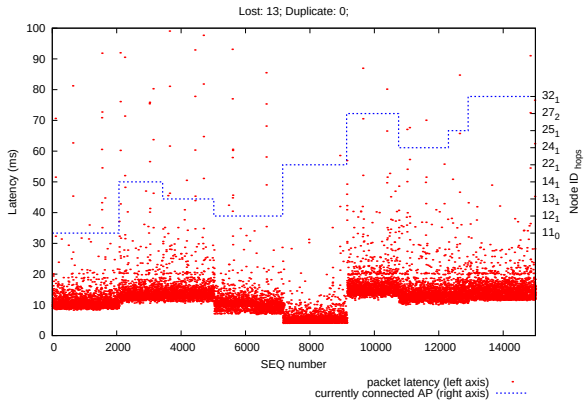


Figure 4.6: P2P test. Latency of packets received at Static Client.

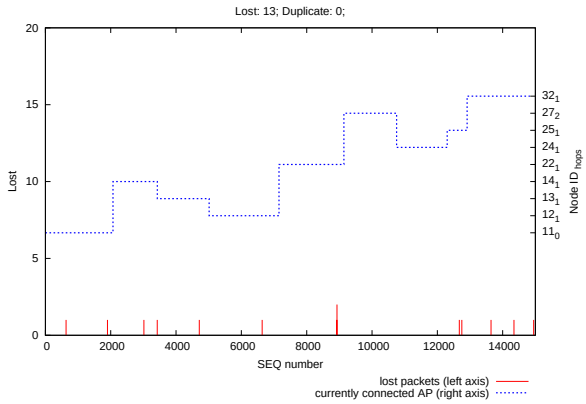


Figure 4.7: P2P Test. Lost packets at Static Client.

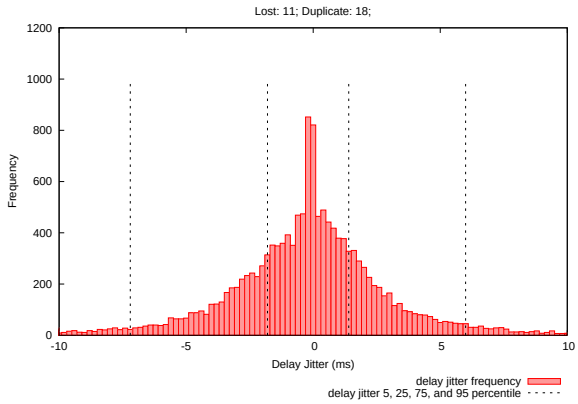


Figure 4.8: P2P test. Delay Jitter for packets received at Mobile Client.

from the Internet as *Sky*. In all experiments we send a full-duplex (two-way) VoIP traffic. The VoIP traffic consisted of 160 byte packets sent every 20 ms at a rate of 64 Kbps, for 5 minutes. We focus our experiments on VoIP as a representative application that poses severe latency requirements.

4.4.2 Measurements

Peer-to-peer UDP test: During this experiment one mobile clients is stationary while the other walks through the previously described path. Routing decisions are based on the

path that decreases the number of wireless hops between the clients in the hybrid wired-wireless overlay network. The stationary Client is connected to node 22 at all times; the Client does not experience any handoff throughout the experiment. Figures 4.5 - 4.8 present the results of this experiment.

In each graph, the access point that serves the mobile client is shown on the right vertical axis. The current access point is represented with a continuous dotted line. Horizontal plateaus of the dotted line represent stable periods in which the access point serves the moving client, while vertical jumps between plateaus represent handoffs between access points. For example, Figure 4.5 shows a transition from node 11 to node 14 around packet number 2000.

Figures 4.5 and 4.6 show the one-way latency of packets as they are received at each client. The initial latency represents 3 wireless hops plus 1 wired hop. This is because there is one wireless hop between the mobile client and node 11, plus one wired hop between node 11 and node 21, plus two wireless hops between node 21 and the stationary client who is connected to node 22. Note that, network wise, this corresponds to one wireless hops. A direct route that did not use the hybrid wired-wireless route would have used an additional wireless hop in order to route packets between the clients.

Around packet 2000, the latency increases slightly as mesh nodes 13 and 14 require one additional wireless hop through the hybrid route towards the stationary client. Around packet 5000, the client connects to mesh node 12. Instead of using the wired-wireless hybrid path, the node uses a direct path as the cost in terms of wireless transmissions is the same. The decrease in latency, which is about 3ms, represents the cost of going through the wired network plus one additional application level router. The two clients connect through the same access point around packet 7000. The mesh node then connects through a node

that is two direct wireless hops away, and then one wireless hop away, until packet 13000. Then, the mobile client moves to a node that resides in a different island, and must use the hybrid path to reach the stationary client. Note that the latency is similar to the one at the beginning of the experiment, where a different overlay link through the wired network was used to forward packets to the stationary client.

Overall, 13 packets were lost in one direction and 11 in the other. Figure 4.7 shows the lost packets at the stationary client, who experienced the most number of losses. Loss is represented as cumulative number of losses over the last 20 packets. A maximum of two consecutive packets was lost around packet 9000. As the wireless medium is shared, a sudden loss may be triggered by a number of factors including external wireless communication or interference from our own wireless network. Also, losses can help to trigger a handoff when in conjunction with the RSSI, the metric of a mesh node that is starting to have better connectivity goes above the threshold. In most real time applications, the effect of a relatively small number of packets being lost can be compensated with no interruption in service or significant quality degradation.

Figure 4.8 present the delay jitter for the stream received at the mobile client. The Inter Quartile Range (IQR), which represents the difference between the 25 and the 75 percentile, was just 3.2ms. This is slightly higher than the IQR experienced during the intra-domain handoff in the previous Chapter, but considering that now both the sender and the receiver are in the wireless network, the increase in the IQR is not significant.

There were only 18 duplicates over the 8 handoffs experienced by the mobile client, 49 packets arrived after $100ms$ out of which 2 packets arrived after $200ms$. The other client did not experienced any handoff, and therefore there were no duplicate packets in this direction.

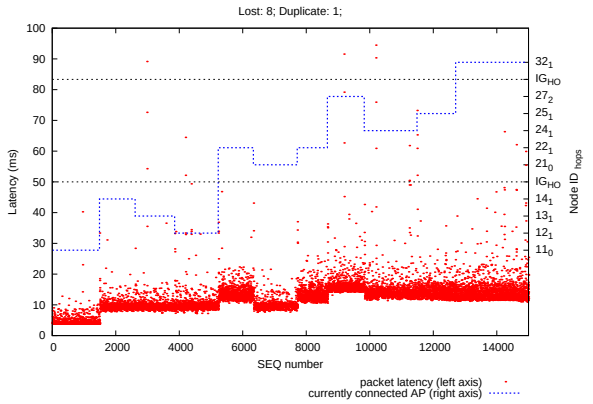


Figure 4.9: Latency. Inter-domain test. Sky is receiver.

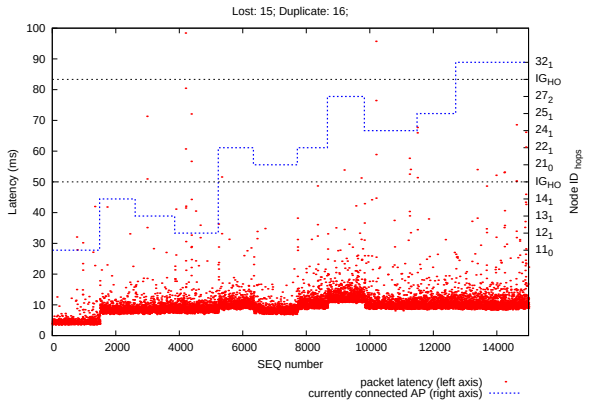


Figure 4.10: Latency. Inter-domain test. Mobile Client is receiver.

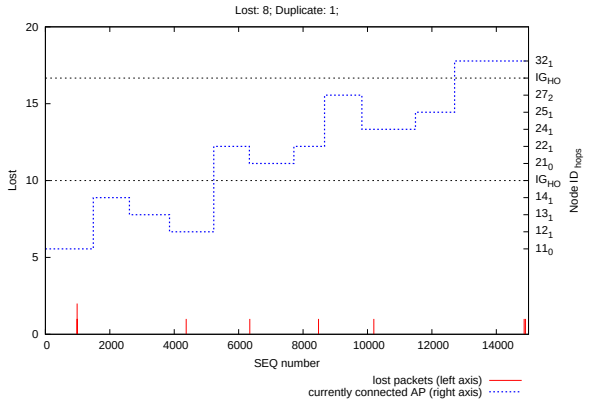


Figure 4.11: Inter-domain test. Sky is receiver. Loss.

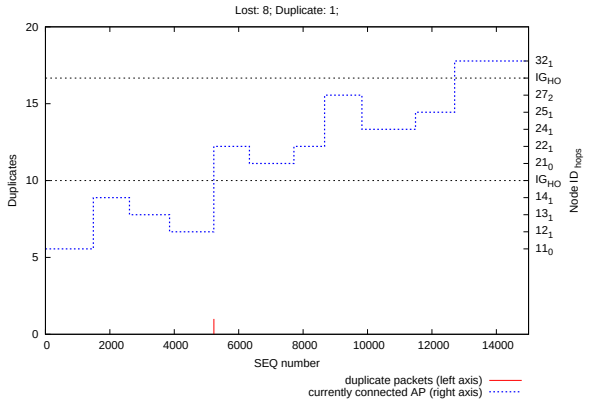


Figure 4.12: Inter-domain test. Sky is the receiver. Duplicates.

Connection Oriented Inter-domain Handoff UDP test: This test is done between a single mobile laptop, *Client*, and the Internet connected machine, *Sky*. Figures 4.9 and 4.10 show the one-way packet latency for packets received at *Client* and *Sky*, respectively. The horizontal lines marked IG_{HO} separate the graph into three areas defined by the Internet gateway forwarding the mobile client's packets to and from the Internet. An inter-domain handoff happens when the dotted line, showing the current access point serving the client, crosses one of the horizontal line.

The initial latency of just about 5ms represents the latency when going through the

Internet gateway that is the owner of the connection. We then move between three different access points, each one hope from the original gateway, and the latency stays constant at around 9ms. The following handoff, around packet 5000, shows the first inter-domain handoff in the system; the new node handling the client, node 22, is closer to a different Internet gateway, node 21. Although the number of wireless hops stayed the same, the latency increases as there is additional processing at the Internet gateways and the wired network needs to be crossed. However, the increase is latency is not symmetrical. The reason is that there is additional overhead in processing packets that flow towards the Internet as they need to be sent to our smesh process an additional time.

Figure 4.11 shows the packets lost at *Sky*. There were only 8 packets lost, but no losses during the inter-domain handoffs. The number of packets that arrived after more than $100ms$ was 2 in the stream from *Sky* to *Client* and 0 in the stream from *Client* to *Sky*. All packets were received within $200ms$. Considering the total number of packets (15000 in each direction), very few packets were lost or delayed.

In Figure 4.12 we show the duplicate packets received by *Sky*. These duplicates are caused by inter-domain handoffs. There was only 1 duplicate packets on the stream in the entire experiment, and they occurred during the first Internet gateway handoff. Since Box 21 was not aware initially whether the packets belong to a new or an already existing connection, it sent the traffic both to the IGMG group and to the final destination (as explained in Section 4.3.3). Because node 11 already had a connection established for that stream in its NAT entries, it forwarded the packets to the Internet destination, and at the same time, it notified the other gateways that it is the owner of the connection, by sending an acknowledgment to the IGMG group. As soon as node 21 received an ownership acknowledgment from node 11, it stopped relaying packets to *Sky* and started forwarding the packets to node

11. Since there was only 1 duplicate packet received by Sky, the inter-domain handoff took less than 20ms to complete. Note that after the notification, all gateways learned about the ownership of that connection. This is the reason there are no duplicates in the second gateway handoff, from node 21 to node 31 that occurs before packet 14000.

TCP handoff test: In the next experiment, we used a 802.11g wireless card in the mobile client, and configured the mesh to 802.11g with a fixed rate of 36Mbps. We moved the *Client* throughout two floors, going down and then up through different stairs in opposite sides of the building.

Figure 4.13 shows the TCP download throughput experienced by the mobile client. Note that we move through a different set of nodes in this experiment. There were 9 handoffs during this experiment. As in the TCP intra-domain experiment where we walk through the same path (depicted in Figure 3.15), the throughput was initially 3Mpbs when the client is connected directly to the Internet gateway, which is lower than expected. This bandwidth is a CPU limited amount; the CPU is 100% utilized at this point. As the number of hops increases, the throughput goes down to about 1Mbps. The inter-domain handoff takes place around second 130, where we see the throughput going up to about 2.3Mbps. The throughput returned back to the original amount when we reached the original location where we started the test. TCP connection remained open at all times, and packets kept flowing regularly.

Mesh Gateway Failure test: It is interesting to see what happens when the Internet gateway used by a TCP connection suddenly fails. If that Internet gateway is the owner of the connection, then we expect that the connection will break. However, if the Internet gateway is not the original owner of the connection, but rather the one closer to the mobile client that forwards packets to the owner Internet gateway, we expect the mesh network to

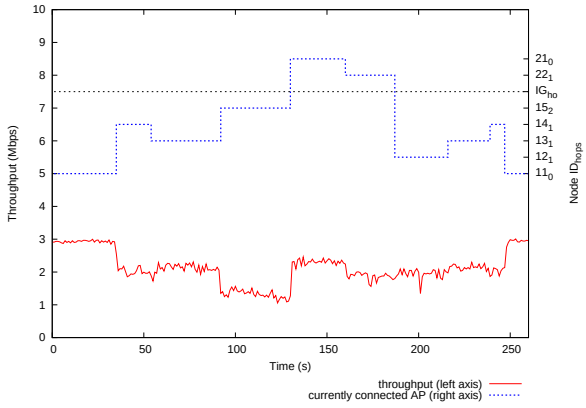


Figure 4.13: TCP Throughput. Multihomed Wireless Mesh. Mobile client is receiver.

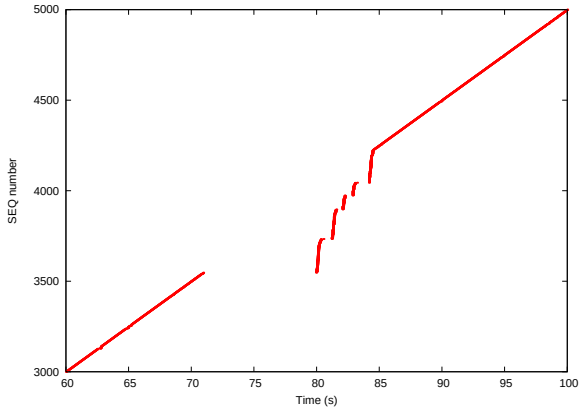


Figure 4.14: TCP fail-over test. Multihomed Wireless Mesh. Sky is the receiver.

discover the failure and adjust the routing such that the data packets will reach the owner gateway.

In this experiment we started a TCP connection between *Client* and *Sky* and then moved the client in the vicinity of a different Internet gateway, forcing a gateway handoff to occur. Then we unplugged the power of the current Internet gateway. Figure 4.14 presents the evolution of a TCP flow where the X axis shows the time and the Y axis shows the packet sequence number. The graph starts after the first handoff from the original gateway. The graph shows about 8 seconds of disconnection required for the mesh network to detect the failure and adjust its routing. After that, it takes a few more seconds for TCP to catch up with the original rate. The network reacting to the failure in a timely manner prevented the disconnection of the TCP connection, overcoming the current Internet gateway crash.

Experiments summary: The experiments show that the SMesh inter-domain protocols provide instantaneous handoff, with a very low overhead caused by messages sent to Internet gateways through the wired network while discovering the originating gateway for a connection. We also show the benefit of multi-homed wireless mesh netorks for lowering the usage of the wireless resource and for increasing the reliability of the mesh. When an

Internet gateway failed and there was at least one other reachable gateway in the mesh, our system was also able to maintain all connection that did not originate from the failed Internet gateway.

As opposed to the wireless intra-domain overhead, which is only dependent on the number of clients, the inter-domain overhead is directly proportional to the number of connections each client has. However, the traffic generated by the inter-domain handoff is small. Considering that the capacity of the wired network is much higher than that of the 802.11 wireless network and that our inter-domain handoff takes less than 40ms in average to complete, we conclude that our inter-domain handoff protocol will not add a significant overhead in the wired network.

Chapter 5

Conclusion

The shift from wired to wireless connectivity has opened the horizon to an era where users expect that their service will not be impaired by their movement between access points. In parallel, real-time applications such as VoIP are expected to keep growing in popularity. We have shown how wireless mesh networks can provide increase coverage and increase redundancy for added reliability, with the steady and stable service necessary to provide such services without any degradation in quality of service to this growing segment.

This thesis presented the architecture and protocols of a seamless wireless mesh network that offers fast intra-domain and inter-domain handoff to mobile users. Our approach allows users to engage in using real-time applications such as interactive Voice over IP without any degradation in quality of service as users move between access points throughout the mesh.

Fast handoff was achieved by using multicast groups to coordinate decisions between access points and between Internet connected access points to seamlessly transfer connections as the mobile clients move throughout the mesh. We also optimized the use of the wireless medium by short-cutting wireless hops through wired connections.

We demonstrated the efficiency of our protocols through live experiments using SMesh [1], a complete and available system. Our approach achieves very good results, allowing unmodified mobile clients to roam freely throughout the wireless coverage area of the mesh network without any interruption in service. We quantified the overhead and demonstrated that it is small compared to the data traffic.

Bibliography

[1] "The SMesh Wireless Mesh Network." [Online]. Available: http://www.smesh.org

[2] "The Spines Overlay Network." [Online]. Available: http://www.spines.org

[3] W. X. Akyildiz, I.F. and W. Wang, "Wireless mesh networks: A survey," *Computer Networks Journal (Elsevier)*, Mar 2005.

[4] S. Akyildiz, I.F.; Jiang Xie; Mohanty, "A survey of mobility management in next-generation all-ip-based wireless systems," *Wireless Communications, IEEE*, vol. 11, no. 4pp, pp. 16–28, Aug 2004.

[5] D. Tang and M. Baker, "Analysis of a Metropolitan-Area Wireless Network," *ACM/Kluwer Wireless Networks. Special issue: Selected Papers from Mobicom'99*, vol. 8, no. 2/3, pp. 107–120, 2002.

[6] B. A. Chambers, "The grid roofnet: a rooftop ad hoc wireless network," Master's thesis, Massachusetts Institue of Technology, May 2002. [Online]. Available: citeseer.ist.psu.edu/chambers02grid.html

[7] J. C. Bicket, D. Aguayo, S. Biswas, and R. Morris, "Architecture and evaluation of an unplanned 802.11b mesh network." in *MOBICOM*, 2005, pp. 31–42.

[8] J. D. Camp, E. W. Knightly, and W. S. Reed, "Developing and deploying multi-hop wireless networks for low-income communities," *Journal of Urban Technology*, vol. 13, no. 3, pp. 129–137, 2008.

[9] "The Champaign-Urbana community wireless network." [Online]. Available: http://www.cuwin.net

[10] "Microsoft research networking research group," http://research.microsoft.com/mesh.

[11] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou, "A multi-radio unification protocol for IEEE 802.11 wireless networks," in *BROADNETS '04: Proceedings of the First International Conference on Broadband Networks (BROADNETS'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 344–354.

[12] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 2004, pp. 114–128.

[13] J. Camp and E. Knightly, "The ieee 802.11s extended service set mesh networking standard," *Communications Magazine, IEEE*, vol. 46, no. 8, pp. 120–126, August 2008.

[14] Y. Bejerano, I. Cidon, and J. S. Naor, "Efficient handoff rerouting algorithms: a competitive on-line algorithmic approach," *IEEE/ACM Trans. Netw.*, vol. 10, no. 6, pp. 749–760, 2002.

[15] C.-F. Chiasserini, "Handovers in Wireless ATM Networks: In-Band Signaling Pro-

tocols and Performance Analysis," *IEEE Transactions on Wireless Communications*, vol. 1, no. 1, Jan 2002.

[16] A. Mishra, M. Shin, and W. Arbaugh, "An empirical analysis of the IEEE 802.11 MAC layer handoff process," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 93–102, 2003.

[17] J.-O. Vatn, "An experimental study of IEEE 802.11b handover performance and its effect on voice traffic," 2003.

[18] I. Ramani and S. Savage, "Syncscan: Practical Fast Handoff for 802.11 Infrastructure Networks," in *Proc. of IEEE INFOCOM*, march 2005.

[19] A. G. Valkó, "Cellular ip: a new approach to internet host mobility," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 1, pp. 50–65, 1999.

[20] R. Ramjee, T. La Porta, S. Thuel, K. Varadhan, and S. Wang, "Hawaii: a domain-based approach for supporting mobility in wide-area wireless networks," *Network Protocols, 1999. (ICNP '99) Proceedings. Seventh International Conference on*, pp. 283–292, Oct.-3 Nov. 1999.

[21] A. Campbell, J. Gomez, S. Kim, C.-Y. Wan, Z. Turanyi, and A. Valko, "Comparison of ip micromobility protocols," *Wireless Communications, IEEE*, vol. 9, no. 1, pp. 72–82, Feb. 2002.

[22] L. DaSilva, G. Morgan, C. Bostian, D. Sweeney, S. Midkiff, J. Reed, C. Thompson, W. Newhall, and B. Woerner, "The resurgence of push-to-talk technologies," *Communications Magazine, IEEE*, vol. 44, no. 1, pp. 48–55, Jan. 2006.

[23] A. Grilo, P. Estrela, and M. Nunes, “Terminal independent mobility for ip (timip),” *Communications Magazine, IEEE*, vol. 39, no. 12, pp. 34–41, Dec 2001.

[24] S. Sharma, N. Zhu, and T. cker Chiueh, “Low-latency mobile ip handoff for infrastructure-mode wireless lans,” *Selected Areas in Communications, IEEE Journal on*, vol. 22, no. 4, pp. 643–652, May 2004.

[25] S. Das, A. Mcauley, A. Dutta, A. Misra, K. Chakraborty, and S. Das, “Idmp: an intradomain mobility management protocol for next-generation wireless networks,” *Wireless Communications, IEEE*, vol. 9, no. 3, pp. 38–, June 2002.

[26] R. Hsieh, Z. G. Zhou, and A. Seneviratne, “S-MIP: A seamless handoff architecture for mobile IP,” in *INFOCOM*, 2003.

[27] K. M. H. Soliman, C. Castelluccia and L. Bellier, “Hierarchical mobile ipv6 mobility management (hmipv6),” June 2004.

[28] R. Caceres and V. N. Padmanabhan, “Fast and Scalable Wireless Handoffs in Support of Mobile Internet Audio,” *ACM Journal on Mobile Networks and Applications*, vol. 3, no. 4, pp. 351–363, 1998.

[29] S. Seshan, H. Balakrishnan, and R. Katz, “Handoffs in Cellular Wireless Networks: The Daedalus Implementation and Experience,” *Kluwer Journal on Wireless Personal Communications, 1996.*, 1996. [Online]. Available: citeseer.ist.psu.edu/115062.html

[30] A. A.-G. Helmy, M. Jaseemuddin, and G. Bhaskara, “Multicast-based mobility: A novel architecture for efficient micromobility,” *IEEE Journal on Selected Areas in Communications*, 2004.

[31] A. Forte and H. Schulzrinne, "Cooperation between stations in wireless networks," *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, pp. 31–40, Oct. 2007.

[32] K. N. Ramachandran, M. M. Buddhikot, G. Chandranmenon, S. Miller, E. M. Belding-royer, and K. C. Almeroth, "On the design and implementation of infrastructure mesh networks," in *in IEEE Workshop on Wireless Mesh Networks (WiMesh)*, 2005.

[33] V. Navda, A. Kashyap, S. Das, "Design and evaluation of imesh: an infrastructure-mode wireless mesh network," in *6th IEEE WoWMoM Symposium,*, June 2005.

[34] S. Ganguly, V. Navda, K. Kim, A. Kashyap, D. Niculescu, R. Izmailov, S. Hong, and S. Das, "Performance optimizations for deploying voip services in mesh networks," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 11, pp. 2147–2158, Nov. 2006.

[35] Y. Amir, C. Danilov, M. Hilsdale, R. Musaloiu-Elefteri, and N. Rivera, "Fast handoff for seamless wireless mesh networks," in *MobiSys 2006: Proceedings of the 4th international conference on Mobile systems, applications and services*. New York, NY, USA: ACM Press, 2006, pp. 83–95.

[36] Y. Amir, C. Danilov, R. Musaloiu-Elefteri, and N. Rivera, "An inter-domain routing protocol for multi-homed wireless mesh networks," *International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2007), Helsinki, Finland*, June 2007.

[37] C. Perkins, "IP Mobility Support," *RFC2002*, Oct 1996.

[38] M. M. Buddhikot, A. Hari, K. Singh, and S. Miller, “Mobilenat: A new technique for mobility across heterogeneous address spaces.” *MONET*, vol. 10, no. 3, pp. 289–302, 2005.

[39] Y. Sun, E. Belding-Royer, and C. Perkins, “Internet connectivity for ad hoc mobile networks,” *International Journal of Wireless Information Networks*, 2002.

[40] Y. Sun and E. M. Belding-Royer, “Application-oriented routing in hybrid wireless,” *ICC ’03*, 2003.

[41] W. Matthew, J. Miller, and N. Vaidya, “A hybrid network implementation to extend infrastructure reach,” *UIUC Technical Report*, 2003. [Online]. Available: citeseer.ist.psu.edu/matthew03hybrid.html

[42] U. Jönsson, F. Alriksson, T. Larsson, P. Johansson, and G. Q. M. Jr., “Mipmanet: mobile ip for mobile ad hoc networks.” in *MobiHoc*, 2000, pp. 75–85.

[43] Y.-C. Tseng, C.-C. Shen, and W.-T. Chen, “Integrating mobile ip with ad hoc networks,” *Computer*, vol. 36, no. 5, pp. 48–55, 2003.

[44] R. K. P Ratanchandani, “A hybrid approach to internet connectivity for mobile ad hoc networks,” *IEEE Wireless Communications and Networking Conference*, 2003.

[45] B. Liu, Z. Liu, and D. Towsley, “On the capacity of hybrid wireless networks,” *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, vol. 2, pp. 1543–1552 vol.2, March-3 April 2003.

[46] R. R. R. Raheleh B. Dilmaghani, Babak Jafarian, “Performance evaluation of rescuemesh: A metro-scale hybrid wireless network,” *WiMesh*, 2005.

[47] H. Wu, C. Qiao, S. De, and O. Tonguz, "Integrated cellular and ad hoc relaying systems: iCAR," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 10, pp. 2105–2115, 2001. [Online]. Available: citeseer.ist.psu.edu/wu01integrated.html

[48] H. Luo, R. Ramjee, P. Sinha, L. E. Li, and S. Lu, "Ucan: a unified cellular and ad-hoc network architecture," in *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2003, pp. 353–367.

[49] I. Akyildiz, J. McNair, J. Ho, H. Uzunalioglu, and W. Wang, "Mobility management in next-generation wireless systems," 1999.

[50] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. of the 18th Symposium on Operating Systems Principles*, Oct. 2001, pp. 131–145.

[51] Y. hua Chu, S. G. Rao, and H. Zhang, "A Case For End System Multicast," in *Proceedings of ACM SIGMETRICS*, Jun. 2000.

[52] Y. Amir and C. Danilov, "Reliable communication in overlay networks," in *Proceedings of the IEEE DSN 2003*, June 2003, pp. 511–520.

[53] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, and A. Terzis, "1-800-OVERLAYS: Using overlay networks to improve VoIP quality," in *Proceedings of the ACM NOSSDAV 2005*, June 2005, pp. 51–56.

[54] R. Droms, "Dynamic Host Configuration Protocol," *RFC2131*, Mar 1997.

[55] K. Egevang and P. Francis, "The IP Network Address Translator (NAT)," *RFC1631*, May 1994.

[56] V. Jacobson, C. Leres, and S. McCanne, "Packet Capture library," http://www.tcpdump.org/.

[57] S. McCanne and V. Jacobson, "The bsd packet filter: a new architecture for user-level packet capture," in *USENIX'93: Proceedings of the USENIX Winter 1993 Conference Proceedings on USENIX Winter 1993 Conference Proceedings*. Berkeley, CA, USA: USENIX Association, 1993, pp. 2–2.

[58] H. Velayos and G. Karlsson, "Techniques to Reduce IEEE 802.11b MAC Layer Handover Time," 2003, kTH Technical Report TRITA-IMIT-LCN R 03:02, ISSN 1651-7717, ISRN KTH/IMIT/LCN/R-03/02–SE, Stockholm, Sweden. April 2003.

[59] D. C. Plummer, "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware," *RFC826*, Nov 1982.

[60] E. Cayirci and I. F. Akyildiz, "User mobility pattern scheme for location update and paging in wireless systems," *IEEE Transactions on Mobile Computing*, vol. 1, no. 3, pp. 236–247, 2002.

[61] "OpenWrt," http://openwrt.org.

[62] C. Demichelis and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)," *RFC 3393*, Nov 2002.

[63] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *In Proceedings of MOBICOM 2003, San Diego*, 2003. [Online]. Available: citeseer.ist.psu.edu/decouto03highthroughput.html

Vita

Nilo Rivera was born in 1974 in Bayamón, Puerto Rico. He received a bachelor degree in mathematics from the Inter American University of Puerto Rico in 1997, and a second bachelor degree in computer engineering from University of Florida in 1999. From 1999 to 2003, he worked developing network protocols at Lucent Technologies. During this time, he obtained a master degree in computer science from the Illinois Institute of Technology. From 2003 to 2008, he worked on his PhD in computer science at The Johns Hopkins University, Baltimore, MD.