Fast Handoff for Seamless Wireless Mesh Networks

Yair Amir, Claudiu Danilov, Michael Hilsdale, Raluca Musăloiu-Elefteri, Nilo Rivera
{yairamir, claudiu, mhilsdale, ralucam, nrivera}@dsn.jhu.edu

Department of Computer Science
Johns Hopkins University
Technical Report CNDS-2005-2 - December 2005
http://www.dsn.jhu.edu

***Abstract*— This paper presents the architecture and protocols of SMesh, a completely transparent wireless mesh system that offers seamless, fast handoff, supporting VoIP and other real-time application traffic for any unmodified 802.11 device. In SMesh, the entire mesh network is seen by the mobile clients as a single, omnipresent access point.**

Fast handoff is achieved by ensuring that each client is served by at least one access point at any time. Mobile clients are handled by a single access point during stable connectivity times. During handoff transitions, SMesh uses more than one access point to handle the moving client. Access points continuously monitor the connectivity quality of any client in their range and efficiently share this information with other access points in the vicinity of that client to coordinate which of them should serve the client.

Experimental results on a fully deployed mesh network consisting of 14 access points demonstrate the effectiveness of the SMesh architecture and its handoff protocol.

I. Introduction

Most wireless network installations today involve a set of access points with overlapping coverage zones, each access point being connected to a wired network tap. Mesh networks remove this strong connectivity requirement 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.

Mobile clients get network access through the mesh by connecting to wireless access points. As a mobile client moves away from an access point and closer to another, it automatically switches its connectivity to the closest access point. This connectivity change involves a transition (*handoff*) of all open connections (e.g. TCP, UDP) so that they use the new access point. 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 [1], [2] 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 [3]. 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 and video conferencing.

This paper presents the architecture and protocols of a completely transparent wireless mesh network system, SMesh, that offers seamless, fast handoff, supporting VoIP and other real-time application traffic. All the handoff and routing logic in SMesh is done solely by the access points, and therefore connectivity is attainable for any 802.11 mobile device that supports DHCP, regardless of its vendor or architecture. The entire mesh network is seen by the mobile clients as a single, omnipresent access point.

Fast handoff is achieved by ensuring that each client is served by at least one access point at any time. 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.

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

SMesh creates a wireless mesh topology in which only some of the access points may be connected to the Internet. Packets are automatically routed through the wireless mesh from mobile clients to the closest access point with a direct Internet connection, and likewise, back to the mobile clients.

SMesh also 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 [4], [5] that provides efficient unicast, anycast, and multicast communication.

The innovative aspects of this paper are:

- The first *seamless* 802.11 wireless mesh network with fast handoff that supports real-time applications such as interactive VoIP and video conferencing.
- Novel use of the DHCP protocol for monitoring connectivity quality of mobile clients and for creating a single, virtual access point throughout the wireless mesh.
- Novel use of multicast for robust mesh Internet gateway to client communication during handoffs, and for localized access point coordination in tracking mobile clients.
- Novel use of anycast for mobile client to mesh Internet gateway communication.

We implemented the SMesh system within the firmware of WRT54G Linksys wireless routers and deployed 14 access points throughout two buildings at the Johns Hopkins University. Tested clients include unmodified Windows XP, Windows Mobile Pocket PC, Linux, and MAC OS X machines.

Our experimental results show that a mobile client that moves throughout a building, switching its access point several times, had no interruption in service, both when sending over UDP and TCP, with handoffs happening instantaneously. During one experiment, two VoIP streams sending 15,000 packets each lost 14 packets combined, and the overhead caused by duplicates due to handoffs was lower than 1.7%. The system experienced 508 duplicates total over 7 handoffs. In addition, we show that when an access point that serves a mobile client experiences a hard crash, the system re-stabilizes and the client regains connectivity within 10 seconds.

The SMesh system is fully functional and available for download at www.smesh.org. The system used for the experiments presented in this paper provides Internet connectivity on a daily basis for the Distributed Systems and Networks Lab at Hopkins. A live view of the mesh topology and the currently connected clients can be seen at www.smesh.org/live.html.

The rest of the paper is organized as follows: Section II describes the architecture of the SMesh system, while Section III presents our fast handoff protocol. Experimental results are presented in Section IV, followed by a comparison with related work in Section V. Section VI concludes the paper.

II. 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. 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 the coverage area served by the wireless mesh.

Following the above goals, we implemented *SMesh*, a system that is capable of providing seamless wireless connectivity to mobile clients. The software architecture of SMesh is shown in Figure 1. Below we describe the two main components of the SMesh architecture: the communication infrastructure and the interface with mobile clients.

A. 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 our Spines messaging system [4], [5]. Spines provides transparent multi-hop unicast, multicast and anycast communication between the wireless mesh nodes. 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 mechanism 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 considered 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.

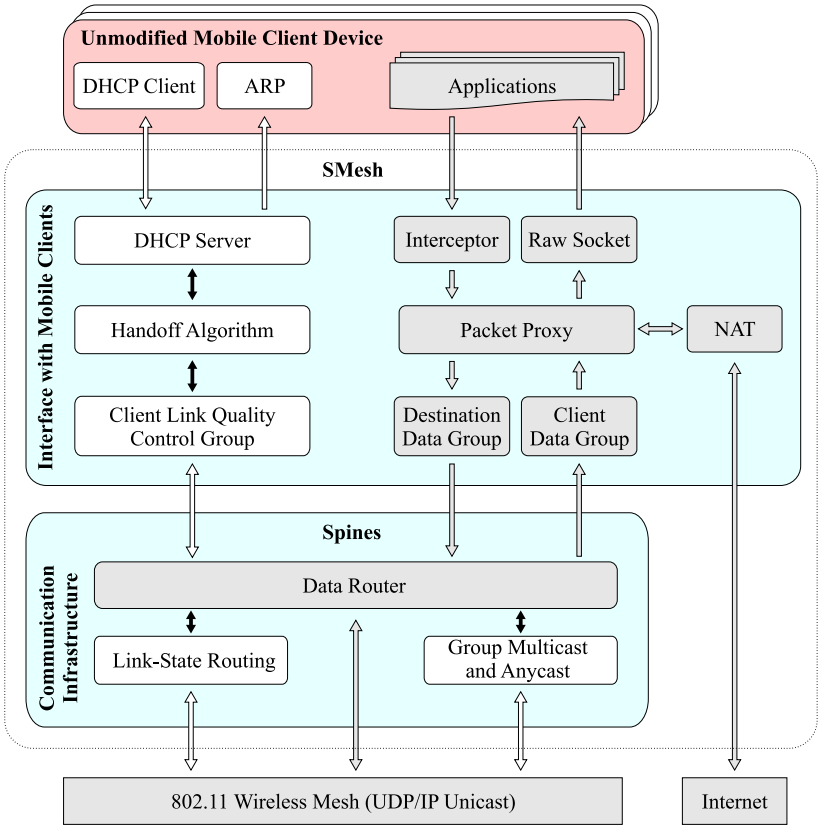

Fig. 1. The SMesh Architecture

Spines has a socket-like interface that makes the interconnection with other components in the SMesh software 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.

Spines allows us to use multicast and anycast functionality in a multi-hop wireless environment. 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 mechanism. Only joins and leaves are flooded to the mesh nodes in the system. The group membership is maintained in Spines in tuples of 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 this paper we only use the number of hops as the routing metric between the wireless mesh nodes.

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 testbed have enough memory to support at least 1000 mobile clients at the same time.

B. Interface with Mobile Clients

SMesh provides the illusion of a single distributed access point to mobile clients. This requires that it provides connectivity information to clients through DHCP and that it routes packets through the wireless mesh network.

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 are 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.

The DHCP Server must force every packet to be routed through SMesh. It sets the default gateway of the client to a generic global gateway and provides a netmask of 255.255.255.254 that forces every packet to route through this gateway.

Of particular importance in the DHCP protocol [6] 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. The DHCP Server inside SMesh, depicted in Figure 1, has the default settings of above options set such that clients are instructed to broadcast (not unicast) DHCP requests every 2 seconds, and to expire their lease only after 90 seconds[1]. This allows SMesh to keep track of the client's location, and on the other hand gives the client enough opportunity to reconnect in case it temporarily goes out of range of any of the mesh nodes, without breaking the exiting connections.

Mesh nodes serve as default gateways for the mobile clients. A Packet Proxy module, depicted in Figure 1, uses an interceptor to grab packets from a client, and a raw socket interface is used to forward packets back to the client.

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, either by the Internet Gateway (for packets coming from the Internet) or by the sending client's 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 forward the original packet to the Internet using Network Address Translation (NAT) [7]. 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. When multiple Internet gateways are used in the mesh, they need to coordinate their NAT tables over their wired connection, so that existing TCP connections continue to flow even if the client moves closer to a different Internet gateway.

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 packet is 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 by TCP at the transport level, and applications using UDP are supposed to handle duplicates).

We describe in the next section how the number of mesh nodes joining the Client Data Group is limited and is equal to one in stable situations, and how it is guaranteed that there is at least one node in the Data Group for each client at any time (unless the client's access point crashes).

III. Fast Handoff Protocol

Traditionally, handoff is provided by using the default DHCP settings. This makes the client broadcast a DHCP request after T_2 seconds (by default, 87% of the Lease time), which allows a different access point to respond and become the default gateway for the client. Even if T_1 and T_2 timers are set to very small values (e.g. 2 seconds), handoff can still take a couple of seconds. Moreover, the client may connect to 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 $100ms$ before being considered lost.

Instead of letting the client "decide" when the handoff should take place by following the DHCP protocol, we make the SMesh nodes track their connectivity to the clients 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 global IP as the default gateway of the client and use gratuitous ARP messages to force handoff to the SMesh node with the best client connectivity.

When 802.11 devices are configured in "infrastructure mode" they inherently do their own scanning for a better access point. 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". This setting is part of the normal setup of any 802.11 device.

We describe below the details of our handoff protocol. These include the link quality metric used by SMesh to

[1]Unfortunately, not all client architectures we tried in our SMesh testbed follow the DHCP RFC to the letter. Therefore, the DHCP Server in SMesh adjusts T1, T2, and Server ID based on the client's MAC address that reveals its architecture. This allows SMesh to enforce the desired client behavior, supporting architectures that are not fully DHCP-compliant.

determine the best access point for each client, the use of multicast groups for managing the clients, and the actual handoff process.

A. Link Quality Metric

We use the DHCP protocol to keep track of the quality of the links to mobile clients. We set the T_1 and T_2 timers to a small value to force the client to broadcast DHCP requests every 2 seconds, as explained previously. Each SMesh node (access point) computes a client link quality measure based on the observed loss of a client's DHCP requests, using a weighted average decay function. Using broadcast instead of unicast for the DHCP requests eliminates the MAC level retransmissions of requests. This way we can evaluate the link quality to each client by the number of DHCP requests received according to the following decay function:

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

where M is the link quality measure and D_f is the decay factor. $Current$ is set to a constant value if the access point received the DHCP request in the previous 2 second interval, and 0 otherwise. The access point calculates this function every 2 seconds for each client in its vicinity. SMesh uses a decay factor of 0.85 to make the protocol resilient to occasional wireless losses of DHCP requests, while maintaining its adaptivity to network condition changes. SMesh uses a $Current$ value of 30 if the access point received the DHCP request to allow integer calculations with discrete mapping. The tie breaker between two access points having the same integer measure (in the range of 0 to 30) is according to the lowest IP of the access point.

The above metric can be further improved by incorporating the Received Signal Strength Indicator (RSSI) of DHCP Requests into our link quality metric. However, there is currently no standard and efficient way of obtaining this parameter in the access points.

B. Client Management

1) Client Multicast Groups: Each client has two multicast groups associated with it. 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 measures 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 a DHCP request 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 measure is shared by the access points periodically by posting it on the client's Control Group. Since only the nodes receiving DHCP requests 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.

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 measure to the client based on link quality measures it receives from other nodes in the client's Control Group.

2) At Least One Semantics: 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).

When a node believes it has the best link quality to a mobile client, it joins its Data Group. When it receives a link quality measure message showing that a different node is better connected, a node in the Data Group issues a *Leave Request*. Leave Requests, sent on the Control Group, are piggy-backed on link quality measure messages. A Leave Request can be acknowledged only by nodes in the *Data Group* that believe they have 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.

A threshold is used in order to decrease oscillations of the Data Group membership. A node joins the Data Group only if its quality measure is higher than the current measure, plus a certain threshold. However, no threshold is used when sending Leave Requests.

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 between one and two seconds during handoffs.

C. Client Handoff

1) Single IP Gateway: 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

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

address to all clients in their DHCP offers (`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 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 to any value (`10.20.30.40` in our experiments), as the communication with the mobile clients is solely based on MAC addresses.

2) Gratuitous ARP: The Address Resolution Protocol [8] allows the mapping of a network layer protocol address to a data link layer hardware address. In 802.11 (and Ethernet) networks, a host can communicate with another host inside the same network only if it knows its hardware MAC address.

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, unicast to the requesting client. All other hosts will discard the ARP request.

A gratuitous ARP is an ARP reply that is not sent as a reply to an ARP request, but rather is broadcasted to the local network voluntarily. Upon receiving such a packet, all hosts will update their ARP caches with the value they received. Typically, gratuitous ARPs are used by hosts to advertise their new hardware address when their network card is changed.

The SMesh handoff mechanism uses gratuitous ARP messages for instantaneous client handoff. 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 (remember that the IP address of the default gateway is set to 10.20.30.40 in our experiment). 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.

The gratuitous ARP is sent by an access point when it joins the Client Data Group and when the access point answers a Leave Request from a different access point. In addition, the gratuitous ARP is sent periodically by the members of the Client Data Group that believe they have the best connection to the client.

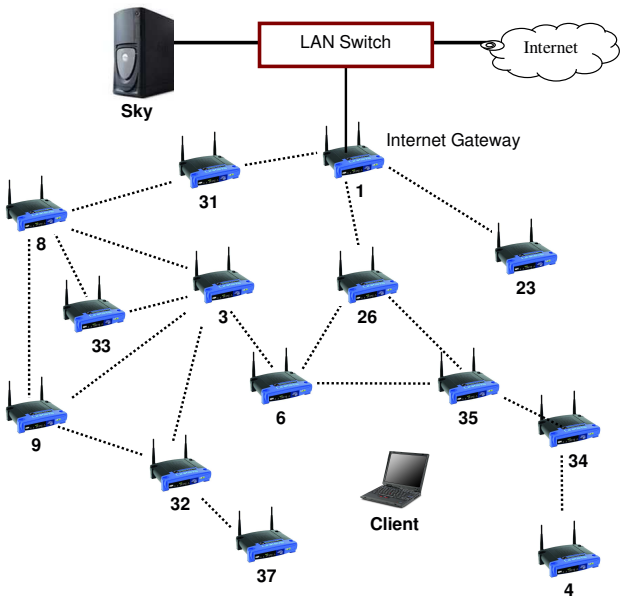

Fig. 2. The SMesh Testbed.

IV. Experimental Results

A. Setup

We deployed SMesh on 14 Linksys WRT54G wireless routers across two buildings at Johns Hopkins University. Only one of the routers was connected to the Internet. As a client we used a laptop computer with an integrated 802.11g card running Windows XP. The topology of the wireless testbed we used in our experiments is shown in Figure 2.

The Linksys routers were modified with the available custom EWRT firmware [9] that provided us with a Linux environment suitable for running the SMesh software. Other than adding SMesh, no other changes were made to the EWRT firmware[3].

The SMesh DHCP Server was set to issue lease times to clients so that they broadcasted DHCP requests every 2 seconds. For the link quality measure we used a $Current$ value of 30, and we set the decaying factor, D_f, to 0.85. The Threshold for joining the Client Data Group was set to 15%. In our experiments these numbers provided the best trade-off between the granularity of the metric and handoff responsiveness.

All our experiments were performed with one mobile client inside SMesh communicating with a Linux machine in the wired network (Internet). 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

[3]The SMesh system was successfully tested to run on regular desktop Linux computers, and also on Linksys WRT54G routers running EWRT [9], Sveasoft [10], and DD-WRT [11] firmwares.

traffic consisted of 160 byte UDP packets sent every $20ms$ at a rate of $64Kbps$. A G.711 encoder/decoder was applied to the VoIP streams.

We first performed a stationary test to set the baseline of our moving experiments. We then proceeded to move across the building starting and ending at the same location as the stationary experiment. We repeated the same experiment, only that instead of sending packets over UDP, we opened a TCP connection between *Client* and *Sky*, and sent packets over this connection at the same rate. Last, 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).

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 with the difference between the clocks at the *Client* and *Sky* machines. The clock difference was measured on a wired LAN, immediately before running each experiment. For VoIP communication it was also important to track 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.

B. Measurements

Stationary client: This test was performed with the mobile client being stationary, in a fixed position for the duration of the entire test. A UDP traffic consisting of $15,000$ packets was sent in each direction: from the Internet box (*Sky*) to the *Client*, and back towards *Sky*. The packet latencies are shown in Figure 3 and Figure 4, respectively. 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 `2H-06` on the right side of the graph refers to node 6 in our topology, which is 2 hops away from the Internet gateway. We notice that even though the client was stationary, its access point changed between the one hop nodes in its vicinity: box `1H-31`, `1H-23` and then `1H-26`. 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), the number of lost packets was 2, and the number of duplicate packets was 172. This amounts to an overhead due to handoffs of 1.14%. During this experiment, 9 packets (0.06% of the total traffic) were delayed by more than $100ms$. As expected, the duplicate traffic occurred only during the handoffs[4].

The reverse stream (Sky is the receiver, Figure 4) did not have any lost or duplicate packets. However, 16 packets (0.1% of the total traffic) arrived later than $100ms$. 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) .

Figure 5 shows the overhead of the our system in comparison with the data traffic. The two lines that overlap at 10 KBytes per second (80Kbps) represent the two data streams (top line is the stream from Sky to the Client). The bandwidth measured with tcpdump is higher than the 64Kbps UDP stream we sent due to IP, UDP, and Spines headers that accumulate on the relatively small (160 byte) packets.

Control traffic from our system is represented as the bottom traffic line. It combines the traffic from Spines (joins and leaves from multicast groups, 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 5 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 send about 19 bytes per second for each client in its vicinity. Clients send a DHCP request of 342 bytes to renew their lease every 2 seconds.

As we can see in Figure 5, a handoff takes place around second 200. The handoff is shown in detail in the upper left of the figure. The first spike of control traffic represents the moment when a node decided to join the Data Group, and sent a join message through Spines. As a consequence, the following data traffic is duplicated, and after 2.5 seconds, the old access point decided to leave the client Data Group (it sends a Leave Request and it immediately receives the acknowledgment: the second spike in the control traffic).

We use the above results as a baseline for the following tests, to provide an idea of our wireless environment.

Moving client: In this test we move the client from the stationary position of the previous experiment, taking it on a 5 minutes trip between three floors inside our building, and ending in the original position. During the test the client changed its access point 7 times, spanning from one-hop away nodes (`1H-23`, `1H-26`, `1H-31`) to three-hops away (`3H-34`).

The latency graphs for each of the two VoIP streams are shown in Figures 6 and 7 respectively.

Each additional hop on the path from the Client to the Internet gateway resulted in an increase in packet latency:

[4]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.

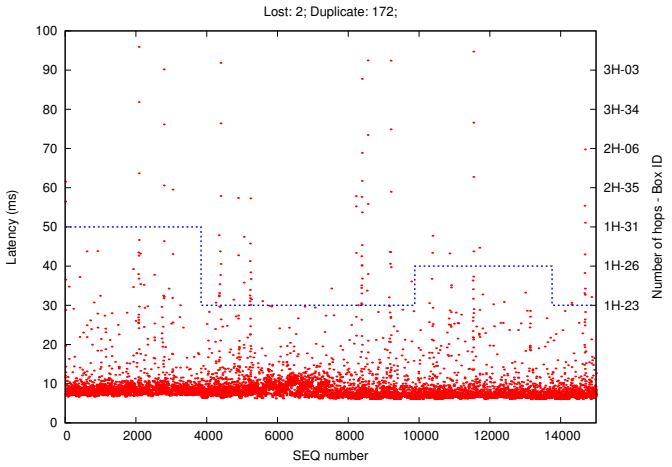

Fig. 3. Stationary client. Client is the receiver.

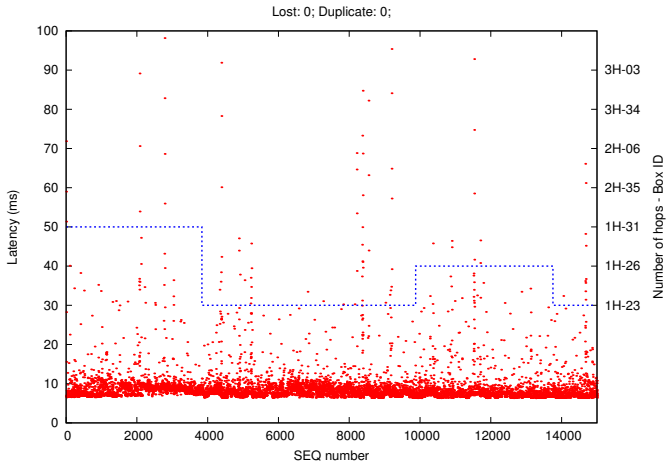

Fig. 4. Stationary client. Sky is the receiver.

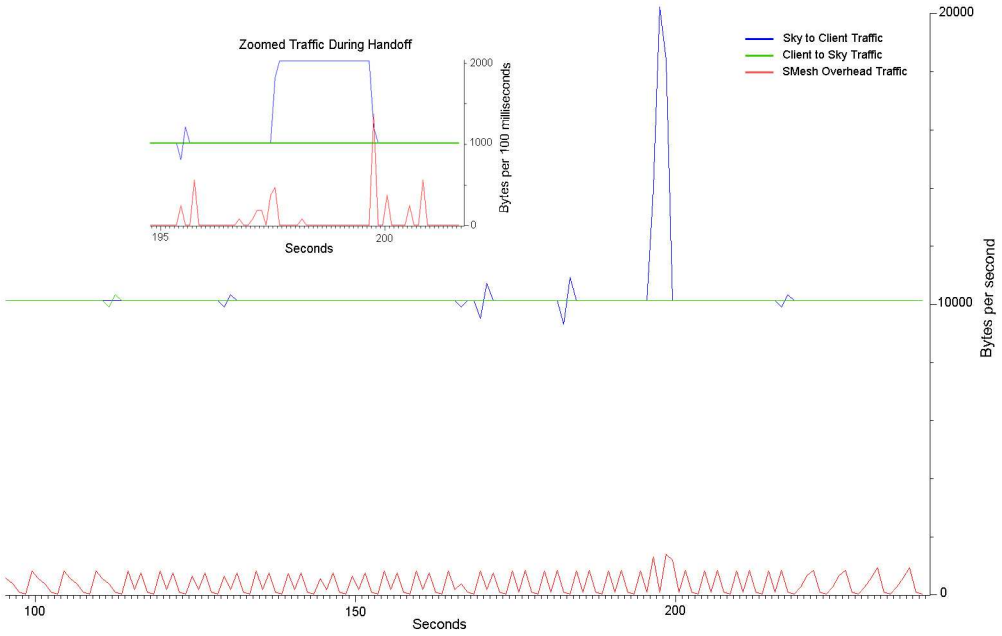

Fig. 5. Stationary client. Traffic overhead. Top lines depict client data traffic. Bottom line depicts SMesh traffic overhead.

between sequence numbers $3,700$ and $4,707$ we were 2 hops away, and between $8,186$ and $11,056$ - 3 hops away. In addition, even when the handoff occurs between nodes with the same number of hops, the packet latency may slightly increase or decrease depending on the quality of the wireless path to the Internet gateway. For example, we see an increase of latency between sequence numbers $1,522$ and $3,405$ in Figure 7.

The data stream towards the *Client* had 12 packets lost, and 508 duplicate packets. As seen in Figure 8, some of the intervals with lost packets are correlated with a handoff, while others others are not. In SMesh, handoffs will occur because of loss of broadcasted DHCP requests (which are even more affected by the loss in the medium as they are not retransmitted by the MAC layer as the unicast data packets). However, a handoff takes place only if there is a better connected access point in the client's vicinity. None of the losses in the experiment was caused by the handoff itself.

As shown in Figure 9, duplicate packets happened only during handoffs. The number of packets that did not arrive within $100ms$ was 55, or 0.36% of the total traffic.

The stream towards *Sky*, depicted in Figure 7 had 2 lost packets, had 0 duplicates and 56 packets that were delayed more than $100ms$.

Figure 10 represents a zoomed sview of the handoff happening at sequence 3700, 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 6).

Our link quality measure relies on loss of broadcast DHCP

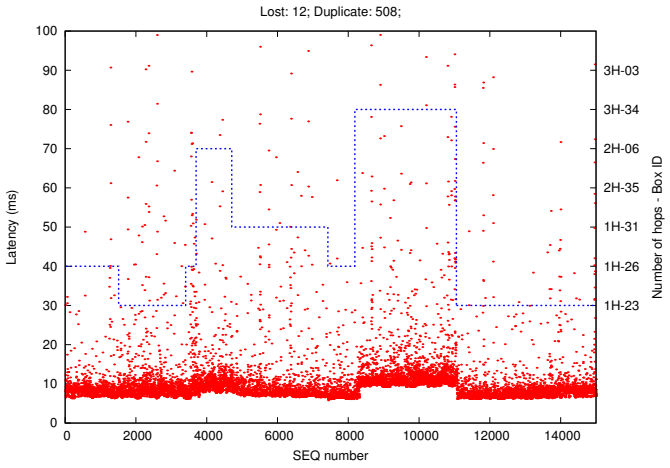

Fig. 6. Moving client. Client is the receiver.

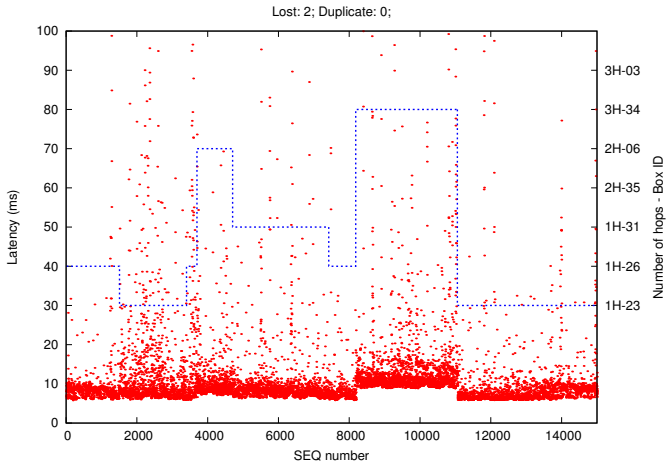

Fig. 7. Moving client. Sky is the receiver.

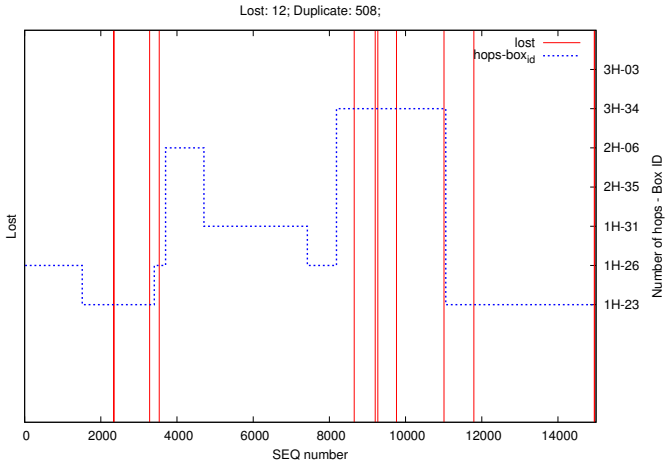

Fig. 8. Moving client. Client is the receiver. Lost packets.

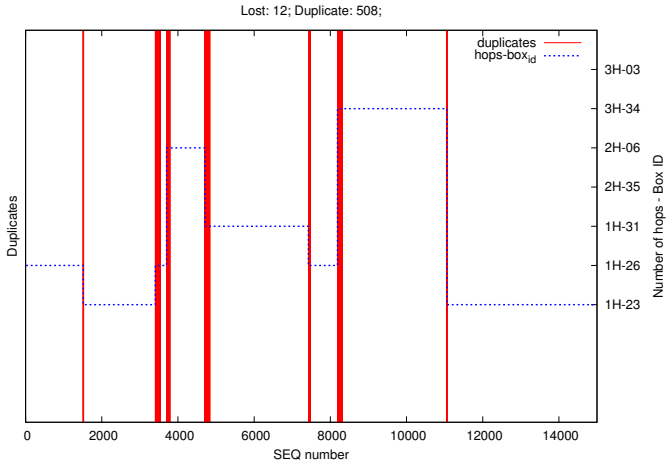

Fig. 9. Moving client. Client is the receiver. Duplicate packets.

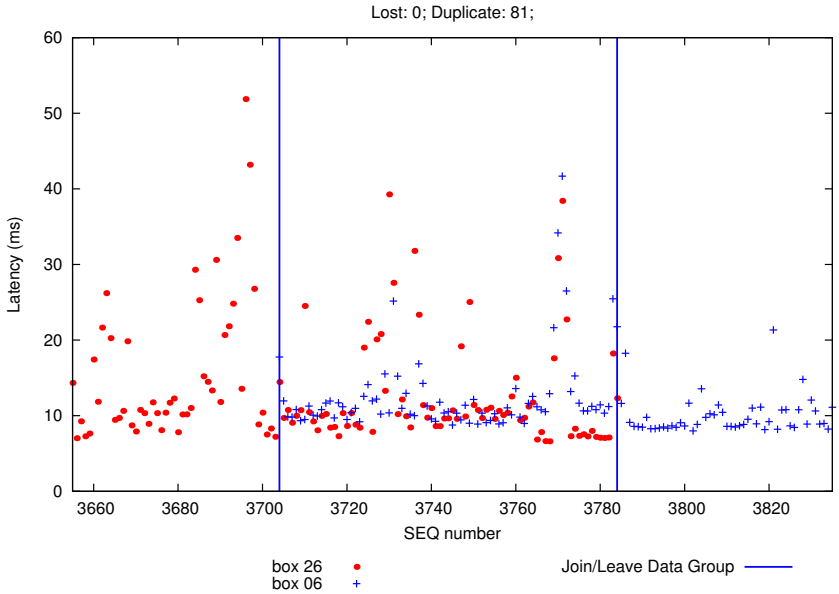

Fig. 10. Moving client. Client is the receiver. Handoff details.

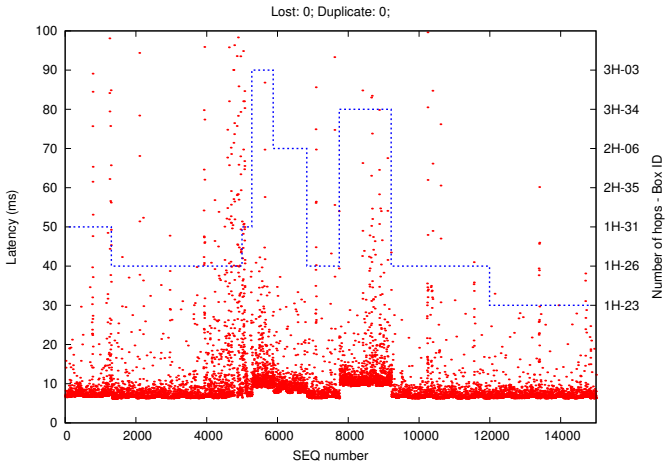

Fig. 11. Moving client. Sky is the receiver. TCP handoff.

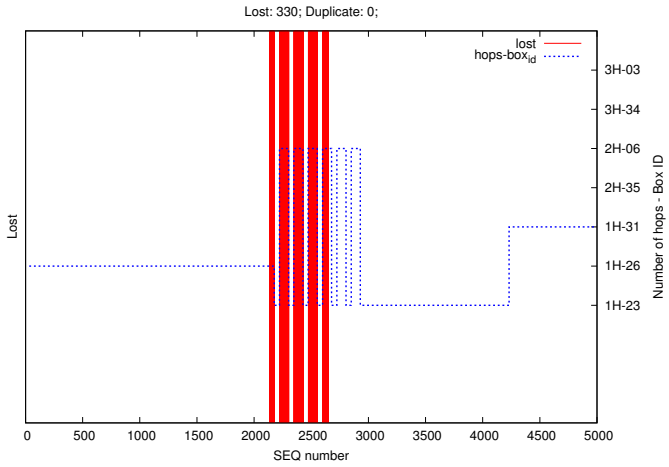

Fig. 12. Sky is the receiver. Topology fail-over. Lost packets.

request packets. As opposed to broadcast packets, unicast packets in 802.11 are retransmitted up to 4 times by the MAC layer and have a higher probability of arriving to their destination. However, if MAC layer retransmissions arrive, they have a larger latency than packets that were not lost. We can see that, just before the handoff, packets arrive with high latencies of up to 50 milliseconds. This latency is mostly due to 802.11 MAC retransmissions from the *Client* to the access point (node 26). Shortly after sequence 3700, a handoff is started, node 6 joins the Client Data Group, and then duplicate packets start to arrive at the *Client*. During this time, handoff negotiation is performed and node 26 leaves the Client Data Group shortly after. We can clearly see that the packet latency after the handoff is lower than before, which means that fewer packets are lost and retransmitted by the MAC layer.

TCP handoff: In the next experiment, we moved the *Client* on the same route, sending the same traffic pattern as in the UDP experiments, only that this time we used TCP as the transport protocol. There were 8 handoffs during this experiment. The TCP connection remained open at all times, and packets kept flowing regularly.

Figure 11 shows the latency of TCP packets. We can see that the latency of packets is slightly higher than that of the UDP packets. We believe this is due to TCP buffering and end-to-end reliable retransmission in TCP. We also note that patterns of high latency (e.g. around Seq. number 4000 or 7000) are immediately followed by a handoff. The high packet latency is caused by losses that are recovered, either at the MAC or at the transport layer.

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 26, sending a VoIP stream to the *Sky* box. As the *Client* was sending packets we suddenly disconnected the power at node 26. Figure 12 shows the packets lost at *Sky* from the *Client* when node 26 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 26 after it fails. Shortly thereafter, node 23 notices it does not receive link quality measures from node 26, and sends a gratuitous ARP to the client, forcing its handoff. In our topology, the minimum hop distance routing selects the route between nodes 6 and 23 to go through nodes 26 and 1. After node 26 crashed, node 6 and node 23 do not receive link quality updates from each other, until routing in Spines is repaired. Therefore, both nodes 23 and 6 believe they have the best link quality to the client. They both insist on taking over the connection from the client, sending to it ARP messages. We can see this behavior in the six handoff oscillations depicted in Figure 12. Since node 6 does not have a route to the Internet gateway until the routing protocol in Spines detects the failure (its original route went through node 26), 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 23 and 6 can reach the Internet gateway). However, it takes a few more seconds for nodes 23 and 6 to send their link quality measures to each other and decide which one should serve the *Client*. Indeed, Figure 12 shows three more handoff between nodes 6 and 23 until 23 is selected to serve the client.

In 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 between access points was minimal. As expected, a short disconnection happens though 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.

V. Related Work

Much of the work on fast 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. Cell networks achieve this 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 [1], [2]. The 802.11 standard does not allow direct application of cell network protocols.

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 [12] in the mid-90s. Ricochet nodes automatically routed client traffic through half-duplex wireless hops until reaching a hardline connection. One of the better known modern mesh networks is the MIT Roofnet project [13], [14] which uses dynamic routing based on link quality measures. Roofnet's emphasis is more on route maintainability and optimization than on handing off a client's connection from one AP to another. Many other community and commercial mesh network implementations also exist [15] [16] [17] [18]. None of them, however, allows for a transparent, fast, handoff.

Microsoft Research has the Mesh Connectivity Layer (MCL) [19] that 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 [20] that using multiple radios on a mesh node combined with smart routing algorithms [21] will dramatically improve the throughput of a wireless mesh network. However, their work necessitates a specific network driver on all mesh network participants, including the clients. Our approach requires no such modification to clients.

Ramani and Savage [22] have recently demonstrated that a quick 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 contrary, our approach uses any unmodified 802.11 client.

Mishra, Shin, and Arbaugh [23] have analyzed the handoff performance in current 802.11b 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 [24] 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 duplicates packets during transition times of the handoff, it masks delays dependent on the vendor of network cards.

Seshan, Balakrishnan, and Katz used a multicast approach in the Daedalus project [25] 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.

VI. Conclusion

This paper presented SMesh, the first seamless 802.11 wireless mesh network with fast handoff that supports real-time applications such as interactive Voice over IP. We introduced a protocol that uses DHCP combined with gratuitous ARP messages for creating a single virtual access point throughout the wireless mesh, supporting any standard 802.11 device. We presented a novel use of multicast to allow coordination between access points in order to guarantee that at least one wireless mesh node manages each client at any time.

The paper demonstrated that in a practical deployment, SMesh 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, and with only occasional losses that were not caused by the handoff process. We quantified the overhead of SMesh and demonstrated that it is small compared to the data traffic.

The SMesh system is already useful, providing network connectivity on a daily basis to all the members of the Distributed Systems and Networks lab at Hopkins, over the mesh topology presented in this paper.

References

[1] Yigal Bejerano, Israel Cidon, and Joseph (Seffi) Naor, "Efficient handoff rerouting algorithms: a competitive on-line algorithmic approach," *IEEE/ACM Trans. Netw.*, vol. 10, no. 6, pp. 749–760, 2002.

[2] Carla-Fabiana Chiasserini, "Handovers in Wireless ATM Networks: In-Band Signaling Protocols and Performance Analysis," *IEEE Transactions on Wireless Communications*, vol. 1, no. 1, Jan 2002.

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

[4] "The Spines Overlay Network," http://www.spines.org.

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

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

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

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

[9] "Portless Networks, EWRT," http://www.portless.net/menu/ewrt.

[10] "Sveasoft Firmware," http://www.sveasoft.com.

[11] "DD-WRT Firmware," http://dd-wrt.com.
[12] Elan Amir and Hari Balakrishnan, "An Evaluation of the Metricom Ricochet Wireless Network," http://nms.lcs.mit.edu/ hari/papers/CS294/paper/paper.html.
[13] B. Chambers, "The grid roofnet: a rooftop ad hoc wireless network," 2002.
[14] John C. Bicket, Daniel Aguayo, Sanjit Biswas, and Robert Morris, "Architecture and evaluation of an unplanned 802.11b mesh network.," in *MOBICOM*, 2005, pp. 31–42.
[15] "Locusworld," http://locustworld.com.
[16] "Tropos networks," http://www.tropos.com.
[17] "Extricom," http://www.extricom.com.
[18] "Cisco," http://www.cisco.com/en/US/netsol/ns621/networking_solutions_package.html.
[19] "Microsoft research networking research group," http://research.microsoft.com/mesh.
[20] Atul Adya, Paramvir Bahl, Jitendra Padhye, Alec Wolman, and Lidong 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, 2004, pp. 344–354, IEEE Computer Society.
[21] Richard Draves, Jitendra Padhye, and Brian 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, 2004, pp. 114–128, ACM Press.
[22] Ishwar Ramani and Stefan Savage, "Syncscan: Practical Fast Handoff for 802.11 Infrastructure Networks," in *Proc. of IEEE INFOCOM*, march 2005.
[23] Arunesh Mishra, Minho Shin, and William 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.
[24] Jon-Olov Vatn, "An experimental study of IEEE 802.11b handover performance and its effect on voice traffic," 2003.
[25] S. Seshan, H. Balakrishnan, and R. Katz, "Handoffs in Cellular Wireless Networks: The Daedalus Implementation and Experience," 1996.