

# DESIGN AND DEMONSTRATION OF LIVE AUDIO AND VIDEO OVER MULTIHOP WIRELESS AD HOC NETWORKS

Yih-Chun Hu

David B. Johnson

Department of Computer Science, Rice University  
Houston, TX 77005-1892 USA

## Abstract

An ad hoc network is a collection of mobile nodes using wireless network interfaces to communicate among themselves, discovering and routing along possibly multihop routes to each other without the assistance of fixed infrastructure. In this paper, we describe the design and demonstration of a set of simple routing protocol mechanisms that substantially improve the transmission of real-time multimedia streams over a multihop wireless mobile ad hoc network. The public demonstration of these mechanisms at the final DARPA GloMo PI meeting showed smooth audio and clear video over the ad hoc network throughout the demonstration, despite frequent routing changes taking place as a car implementing one endpoint node moved continuously. We implemented our routing extensions in the Dynamic Source Routing protocol (DSR) running over standard 2 Mbps Lucent WaveLAN-II radios; the two endpoint nodes ran standard Microsoft Windows NetMeeting, communicating entirely through normal IP packets over our 8-node DSR ad hoc network.

## I. INTRODUCTION

Ad hoc networks have the potential to provide effective communication services for groups of wireless mobile users, without requiring the aid of any centralized administration or fixed infrastructure such as base stations or access points. In an ad hoc network, nodes automatically establish routing among themselves. Each node in an ad hoc network acts not only as a host but also as a router, forwarding packets for other nodes; as nodes move or as wireless propagation conditions change, wireless links between nodes may break, and the routing protocol in the ad hoc network automatically adapts the routing to these changes in network topology. The resulting “multihop” routing allows nodes not within direct wireless transmission range of each other to communicate. In addition to circumstances where there is no existing network infrastructure, ad hoc networks are also useful where the existing infrastructure cannot be used for reasons such as security, service level requirements, or usage cost.

Some applications of ad hoc networks require the ability to support real-time multimedia streams such as live audio and video over the network. Examples of such applications include video conferencing, remote site monitoring, and unmanned vehicle operation. However, such support in an ad hoc network is particularly challenging due to the rapid changes in routing that may occur with node motion and due to occurrences of interference and congestion in the shared radio spectrum used by the nodes. Although a number of mechanisms for providing the necessary Quality of Service (QoS) support for such applications in ad hoc networks have been proposed, these mechanisms introduce complexity into the system and generally increase system overhead. In addition, published reports of

testbed implementations or demonstrations of ad hoc networks [1, 23] do not provide mechanisms supporting these types of real-time multimedia streams.

In this paper, we report on the design and demonstration of a set of three mechanisms we added to the *Dynamic Source Routing* protocol (DSR) for ad hoc networks [14, 16] to support live audio and video streams. DSR is a simple and efficient protocol for routing in ad hoc networks, which has been previously demonstrated through simulation and testbed implementation to perform well [5, 13, 22, 24]. Our goal in this work was to develop a set of lightweight extensions for DSR that perform well and that preserve the basic operation of the DSR protocol. These extensions can also be applied to other similar ad hoc network routing protocols such as AODV [28].

Our demonstration at the final DARPA Global Mobile Information Systems (GloMo) Principal Investigators Meeting in July 2000 showed smooth audio and clear video performance over a continuously moving multihop wireless ad hoc network, using off-the-shelf Microsoft Windows NetMeeting [26] audio and video software. Since our implementation and demonstration of these mechanisms at the GloMo PI meeting, some simulation results have been shown for these and similar mechanisms [8, 10], but we report here on the first and we believe currently only implementation of these mechanisms; this work thus represents the only experimental results showing the effectiveness of these mechanisms in a real mobile ad hoc network.

## II. SUMMARY OF THE DSR PROTOCOL

We developed our support for live audio and video streams over ad hoc networks based on the Dynamic Source Routing protocol (DSR) [14, 15, 16]. In this section, we provide a brief summary of the basic operation of DSR. Complete details on DSR are available in the current protocol specification [17].

In order to reduce network overhead and to provide quick reaction on necessary routing changes in the network, DSR is a *reactive* routing protocol and operates entirely *on demand* [22]. That is, unlike traditional *proactive* routing protocols that exchange routing information between routers (e.g., periodically) so that each node should always have up-to-date routes to all destinations in the network, a node using DSR only attempts to discover a new route to some destination when necessary for routing a packet to that destination. DSR also attempts to detect broken wireless links between nodes along that route only when sending a packet along that route. This on-demand operation removes all background routing overhead packets present in other

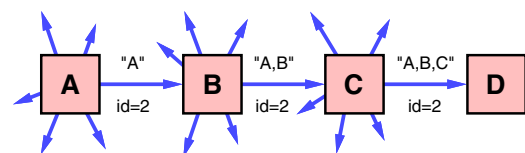


Fig. 1. Example of DSR Route Discovery

This work was supported in part by the Air Force Materiel Command (AFMC) under DARPA contract number F19628-96-C-0061 at Carnegie Mellon University, by NASA under grant NAG3-2534, and by a gift from Schlumberger. Yih-Chun Hu was also supported by an NSF Graduate Fellowship. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of AFMC, DARPA, NASA, Schlumberger, Rice University, Carnegie Mellon University, or the U.S. Government or any of its agencies.

protocols, and allows the routing overhead to scale automatically as the need to react to relevant new topology changes in the network increases.

New routes in DSR are discovered through a procedure known as *Route Discovery*, as illustrated in Figure 1. The effect of Route Discovery is to actively search through the network for a route to the intended destination. Each node maintains a *Route Cache* of routing information that it has learned; when some node (such as node  $S$  in Figure 1) originates a packet for some destination (such as node  $D$ ), if the sender has no cached route to the destination, it initiates Route Discovery by transmitting a ROUTE REQUEST as a local broadcast packet.

A ROUTE REQUEST packet contains four fields: the address of the node initiating the request, the address of the target of the request (the intended destination node to which this sender is searching for a route), an identifier unique with respect to other Route Discovery attempts this source has initiated for this target node, and a record of the route discovered so far by this ROUTE REQUEST. When a node receives a ROUTE REQUEST, if it is not the target of this Route Discovery attempt, the node checks the unique identifier to determine whether or not it has previously received another REQUEST belonging to this same Route Discovery attempt. If it has, the node silently discards the REQUEST; otherwise, the node appends its own address to the route recorded in the REQUEST and locally rebroadcasts the REQUEST. When the REQUEST reaches the target node for this Route Discovery, the target node sends a ROUTE REPLY packet to the initiator of the ROUTE REQUEST, including in the REPLY the recorded route from the REQUEST. A number of useful optimizations to this basic Route Discovery procedure are used to reduce the overall overhead of Route Discovery [17].

When sending a packet along some route, DSR uses a procedure known as *Route Maintenance* to detect when some link along this route has broken, causing this route to no longer work to reach the intended destination node. In Route Maintenance, the node sending a packet and each node forwarding the packet along the source route attempt to verify that the packet reached the next-hop node along the route.

Such a node can obtain this verification for Route Maintenance in a number of possible ways. In particular, many wireless links provide link-level acknowledgement of each packet transmitted; these acknowledgements can be used by Route Maintenance with no extra overhead. If link-level acknowledgements are not available, a *passive acknowledgement* [18] may be used, in which this node overhears the wireless transmission of the next-hop node forwarding the packet on to the following hop along the source route. Finally, if neither of these methods is available, this node may request the next-hop node to return an explicit, network-layer acknowledgement when it receives this packet over the link.

If an acknowledgement is not received after possibly several retransmissions over this link, this node returns a ROUTE ERROR packet to the source of the packet. For example, in a route from source node  $A$  to destination  $D$ , if the previous hop before node  $D$ , node  $C$ , has been unable to receive an acknowledgement from  $D$  for some packet, and thus determines that the link from  $C$  to  $D$  is broken, node  $C$  returns a ROUTE ERROR to node  $A$  identifying this broken link. When node  $A$  receives the ROUTE ERROR, it removes the specified link from its Route Cache so that in the future it does not use a route containing that link. If node  $A$  has another route to  $D$  in its Route Cache, it may use this route for any subsequent packets it sends to  $D$ , or it may reinvoke Route Discovery to attempt to discover a new route to  $D$ .

### III. SUPPORTING LIVE AUDIO AND VIDEO

#### A. Preemptive Route Maintenance

The standard DSR Route Discovery and Route Maintenance procedures work very well for most types of data, but when a route in use breaks, some latency is introduced before the data can begin flowing again over a new route. Route Maintenance must attempt a number of retransmissions over the broken link before sending a ROUTE ERROR to the source of the data, and Route Discovery requires a network round-trip between the source node and the target node to discover a new route to the target. Although the total of this added latency when a route breaks is small (typically less than 100 ms total, with our hardware), its effect on real-time multimedia streams such as live audio and video is undesirable.

To avoid this latency, we used the measured signal-to-noise ratio (SNR) for received packets to detect when a route in use is likely to break soon. For the wireless LAN cards used by the nodes in our implementation (IEEE 802.11 Lucent WaveLAN), for each received packet, the card reports to the network interface device driver software the measured signal strength and noise values along with the contents of the packet; this is a common feature supported by most commercial wireless LAN cards and other radio devices. The term *preemptive Route Maintenance* refers to a number of possible approaches that make use of this SNR information to detect the likely imminent failure of a link, in order to use an alternate route if already known, or to be able to initiate a new Route Discovery in time to discover a new route before the old route actually breaks.

One approach to preemptive Route Maintenance is for each node to keep statistics of the SNR for recent packets it has received from each of its neighbors. When the node receives a new packet with an SNR below some threshold, it may choose to send a warning of this to the original sender of the packet. For example, if this packet and the last several packets received from this neighbor have all been below the threshold, it is likely that the link from that neighbor will soon break. To send the warning to the sender of the packet, this node could send a special type of ROUTE ERROR as a warning, indicating that the link may fail soon. A node receiving such a notification may continue to use the specified link, but if it is routing real-time multimedia packets over routes including that link, it should initiate a new Route Discovery to find a more suitable route soon.

A number of additional sources of SNR information are also available for use with preemptive Route Maintenance. For example, if a node makes use of passive acknowledgements for Route Maintenance, the SNR of a received passive acknowledgement from the next-hop node forwarding a packet transmitted by this node, can serve as an indication of the quality of the link from this node to that next-hop node. Furthermore, the measured SNR for any other packet overheard by this node (e.g., a broadcast packet or any packet received by this node while operating its network interface in “promiscuous” mode) may also provide an indication of the quality of the link between this node and the node transmitting that packet, if the transmit power used is known.

In some environments, it may be important to distinguish the meaning of these SNR measurements for the direction of transmission over the link. In particular, due to effects such as differing sources of wireless interference or possible differences in the radio and antenna hardware at both ends of a wireless link, wireless propagation may not work equally well in both directions between two nodes. For the SNR of a packet received by this node for forwarding along a route, the measured value directly indicates the quality of the link being used to forward this packet and future packets along the same route. For the SNR of a received passive acknowledgement, however, the measured value only directly indicates

the quality of the link *from* this next-hop node *to* this node, for example as might be used in a reverse route used for end-to-end communication in the opposite direction between the same two end nodes (e.g., for the return of an end-to-end TCP acknowledgement packet). Similarly, the SNR value measured for any other packet overheard by this node only directly indicates the quality of the link *from* the node transmitting this packet *to* this node. In each of these cases, though, these SNR values also may be useful indirectly in helping to assess the quality of the link for future packets transmitted to that neighbor node by this node, especially when the underlying MAC layer requires bidirectional communication, as in IEEE 802.11 [12].

### B. Using SNR to Limit Route Discovery

The ability of the network interface hardware to measure the signal-to-noise ratio (SNR) for each received packet, as described above in Section III-A, can also be used to improve the behavior of the Route Discovery process for real-time multimedia streams such as live audio and video. Due to natural fluctuations in received signal strength from effects such as multipath propagation, some packets may be able to be received by nodes much further away from the transmitting node than is typical for packets transmitted by that node.

This phenomenon creates a potential problem for Route Discovery in that a ROUTE REQUEST packet may sometimes be received by a node that would not be able to receive most other packets transmitted by that same node. If this ROUTE REQUEST is processed normally by that node as part of this Route Discovery attempt, the discovered route may be very unreliable, since this node may not be able to receive most data packets sent along that route to it or through it as an intermediate node.

In addition, in the same way as preemptive Route Maintenance attempts to find alternate routes when a link in a route in use falls below the SNR threshold, Route Discovery should avoid finding such routes, if possible. That is, if a route is discovered by Route Discovery in which one or more links have a very low SNR, this may be an indication that such links in the route are likely to break soon; in this case, it may be much more efficient and cause much less disturbance to real-time multimedia streams over this route, for Route Discovery to be able to find a different route instead.

To address these problems, the processing of a Route Discovery when attempting to find a route for use by a multimedia stream could be modified to limit the spread of the ROUTE REQUESTS based on SNR. In this case, a node that receives a ROUTE REQUEST such that the SNR of this received packet falls below the threshold should not forward the REQUEST. Thus, all routes discovered will consist entirely of links with adequate SNR at the time that the route is discovered.

### C. Per-Hop Flow State Maintenance

Audio and video applications tend to send very small packets to minimize the buffering latency at the transmitter. Unfortunately, this means that the source route normally present in the header of each DSR packet could represent considerable total overhead. In addition, we would like to be able to differentiate packets belonging to real-time multimedia flows and ordinary data packets.

We have designed a general mechanism called *flow state* to support these goals [10]. We provide a summary of the basic flow state mechanism here. With flow state, most DSR packets do not contain a source route header. Once a node sending a packet to some destination has discovered a source route to that destination, the node sends the packet as a normal DSR packet containing the full source route header. As the packet is forwarded

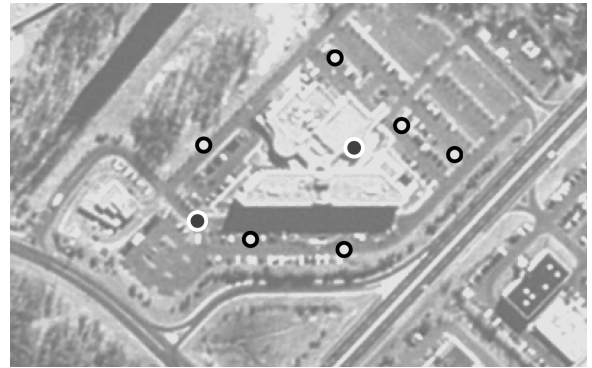


Fig. 2. Location of nodes in the demonstration

to the destination based on this source route, the flow state mechanism allows each node forwarding the packet to remember the address of the next hop along this source route. Subsequent packets from this sender may then be forwarded along the same route to this destination with no source routing information present in packet header of any of these packets. The flow state established at each hop along the route is “soft state” and thus automatically expires when no longer needed.

Our current design of the flow state mechanism [10] is more general than what we used in our demonstration, but for the use to which we put it here, the two versions are functionally the same. Each node maintains a Flow Table, which associates a (*source address, flow identifier*) pair with a full source route to the destination for that flow. To establish Flow Table entries, packets are sent with a source route as normal, except that a flow identifier is also included. A node receiving such a packet adds the new flow information to its Flow Table.

To send a packet along that flow once all intermediate nodes have associated the flow identifier with the source route, the sending node replaces the destination address in the packet’s standard IP header with a special encoding of the flow identifier for that flow. If a node receives a packet for forwarding without a source route in the packet’s header, the node forwards the packet to the next hop indicated in this node’s Flow Table entry for that flow identifier. If, however, the node has no corresponding Flow Table entry, it returns a special form of ROUTE ERROR packet to the source node; the source then reestablishes the flow state as when initially beginning to use that route. The flow state also allows a forwarding node to differentiate real-time multimedia packets from ordinary data packets, as this state can also be represented in the Flow Table entry.

## IV. GLOMO DEMONSTRATION

### A. Demonstration Design and Configuration

In our demonstration ad hoc network at the GloMo PI meeting, we used Microsoft Windows NetMeeting [26] to provide live audio and video between a stationary node and a moving car over multiple wireless hops. The demonstration was performed at the Sheraton hotel in Eatontown, New Jersey, a large 6-story hotel surrounded by parking lot on all sides; Figure 2 illustrates the configuration of the nodes in our ad hoc network. The stationary endpoint node was located in the hotel as shown on the right in Figure 2, with the moving car endpoint (shown on the left) driving in the hotel parking lot, continuously circling the hotel building (counterclockwise) without stopping. All routing in the ad hoc network was done using the Dynamic Source Routing protocol (DSR) integrated with the extensions for real-time multimedia support described in Section III.

The ad hoc network allowed these two endpoint nodes to remain con-

nected as the moving car endpoint circled the hotel. Each of the other six nodes in the ad hoc network shown in Figure 2 was implemented as a car parked in the hotel parking lot, with a FreeBSD Unix laptop in each car implementing DSR. Although these other nodes were stationary during the demonstration, the multihop wireless ad hoc network route between the stationary hotel endpoint node and the moving car endpoint changed rapidly throughout the demonstration. When the moving car endpoint node was near the hotel endpoint, the best route between these two nodes was a direct one-hop DSR route. As the moving car passed this point in circling the hotel, the route continued to change to incorporate more of the intermediate nodes, until the car reached the halfway point around the hotel. At this point, the best route between the moving car and the hotel endpoint switched to route around the hotel building in the opposite (shorter) direction, and this route then became progressively shorter as the moving car returned to near the hotel endpoint on its way again around the building. Keeping the intermediate nodes stationary also simplified the operation of the demonstration within the crowded hotel parking lot.

As mentioned, the two endpoint nodes in our demonstration (the moving car and the stationary node in the hotel) ran Microsoft Windows and NetMeeting. All application and operating system software on these two nodes were the standard, unmodified commercial version, and all communication was through standard IP packets. The DSR protocol and our implementation of it are thus entirely compatible with this commercial off-the-shelf software.

All wireless links in the ad hoc network were implemented using Lucent WaveLAN-II wireless LAN PCMCIA cards [19]. The radios support a maximum bitrate of 2 Mbps and are compatible with the IEEE 802.11 wireless LAN standard [12]; they operate in the 2.4 GHz ISM band with a transmitter power level of 15 dBm (30 mW). In setting up the network, we measured signal strength and packet transmission reliability between the stationary cars, and placed these nodes such that each was normally able to communicate only with its directly adjacent nodes around the hotel building.

In order to provide DSR routing functionality at the two endpoint nodes, we used an additional laptop at each endpoint. At the moving car endpoint node, one laptop ran Microsoft Windows and NetMeeting, and the other ran FreeBSD Unix and DSR. The two laptops were connected by a short wired Ethernet segment, using static IP routing configuration on these two laptops: the Windows laptop's IP default router was the FreeBSD laptop, and the FreeBSD laptop was configured with a static route to the Windows laptop. The stationary endpoint in the hotel was configured in the same way, except that the connection between the two in this case used a 2 Mbps point-to-point wireless link based on Lucent WaveLAN-II hardware with Yagi directional antennas. This link was wireless rather than wired as in the moving car endpoint, in order to extend the signal outside the hotel into the parking lot where the ad hoc network operated; we used a different IEEE 802.11 frequency channel for this link than used in the ad hoc network nodes, so that this link was isolated to the two laptops implementing the stationary hotel endpoint.

### B. Protocol Implementation

We implemented the DSR protocol extensions described in Section III by modifying our existing DSR implementation in the FreeBSD Unix kernel. FreeBSD is a freely available open-source version of Unix based on the Berkeley 4.4BSD-Lite Release 2 Unix distribution. In total, we changed approximately 1600 lines of source code in our DSR kernel to support these changes.

For both preemptive Route Maintenance (Section III-A) and using SNR in Route Discovery (Section III-B), we needed a way to get the necessary SNR information out of the network interface device driver into the DSR routing code. When a packet is received, the device driver places the packet in a kernel memory buffer known as an *mbuf* [25]. Since the design of our extensions depend on the SNR of a received packet only in order to check if it is below threshold, we simply added a new flag to the mbuf header to indicate that the packet in this mbuf was received with low SNR. The use of this flag avoids major changes to the format of packets in mbufs and significantly reduces any compatibility concerns for this functionality with the rest of the protocol handling source code.

In implementing preemptive Route Maintenance, we took advantage of the node configuration used for our demonstration. Since we had only one moving node, we implemented preemptive Route Maintenance only for that node. When the SNR to the next hop for the moving node was fading, this node could then initiate a new Route Discovery itself, eliminating the need to actually send a special ROUTE ERROR packet when the SNR dropped below the threshold. In addition, in our implementation, the node reacted when just one packet fell below the SNR threshold. Though this approach may result in a number of unnecessary Route Discoveries, it reacts quickly to degrading link quality.

In implementing the flow state extensions to DSR (Section III-C), we added a Flow Table to each node to track all of the flows originating at or being forwarded by that node. When a node forwards a packet using a source route, it checks to see if the packet also contains an option specifying a flow identifier for this flow. If so, the node adds the route to its Flow Table. At the sender, we added an application program that allowed real-time multimedia flows to be identified, using a flow specification similar to RSVP [4]. An entry was added to the Flow Table for each flow identified in this way. When a sending node begins to use a new route for some flow in its Flow Table (e.g., when first using the flow or after the route in use is changed due to Route Maintenance), for the next ten packets that the node sends along that flow, the node includes a normal full DSR source route header and the new flow identifier in each packet; all other packets were sent using only a flow identifier. Repeating the full source route on each of these initial packets helps ensure that all of the nodes along the route have received the new routing information and associated it with the new flow identifier. If a node receives a packet for forwarding without a source route in the packet, but this node has no corresponding Flow Table entry (e.g., because it did not receive the earlier packet establishing that flow), the node returns a special ROUTE ERROR to the source, which then reestablishes the flow state, as described in Section III-C.

To send a packet using a flow identifier, we used an encoding of the flow identifier in the packet header that allowed such packets to carry no per-packet overhead for DSR. Each flow identifier is 16 bits long. In the packet header, we represented the flow identifier in the packet's IP Destination Address field as an address of the form 127.0.xxx.yyy, where xxx.yyy is the flow identifier. Such addresses are normally reserved for the loopback network interface [29] and thus do not otherwise appear in the Destination Address field of a packet sent over the network. Since each node in our ad hoc network supported DSR and flow state, this address could in our implementation be recognized as a special case. Alternatively, the flow identifier could be represented in a special small header in the packet [10], but in our implementation, we opted for the representation in the Destination Address to further reduce overhead for carrying the audio and video streams.

Finally, in this implementation, we modified the FreeBSD WaveLAN device driver and our DSR code to allow DSR Route Maintenance to uti-

lize the link-layer acknowledgements built into the IEEE 802.11 MAC protocol. Specifically, after completing transmission (and any link-layer retransmission attempts) for a packet, the device driver calls a new procedure within Route Maintenance to indicate the success (802.11 acknowledgement received) or failure (no acknowledgement received) for that packet transmission; if a transmission failure is indicated, DSR's Route Maintenance reacts by sending a ROUTE ERROR as appropriate. This ability to use link-layer feedback is a part of DSR's design [14] but to our knowledge has not been implemented before on 802.11 for DSR or other routing protocols for ad hoc networks. Since this acknowledgement is already required for all unicast IEEE 802.11 transmissions, DSR is then able to perform Route Maintenance with no extra overhead.

### C. Demonstration Results

We set up the ad hoc network for the demonstration and tested it over one day and demonstrated it for different groups of people at the meeting frequently over the following two days. In each run of the demonstration, we operated the ad hoc network continuously, with most runs lasting several hours or more; during each run, the moving car continued to drive around the hotel building, with brief breaks only to change drivers of the car, without resetting the laptops or restarting the network. The routing between the stationary endpoint node in the hotel and the opposite endpoint node in the moving car thus continued to change frequently during the operation of the demonstration. In all runs of the demonstration, all aspects of the protocols and extensions for live audio and video support worked extremely well in almost all respects.

Observers of the demonstration in the hotel could converse with the driver of the car and could see the view from a camera in the car pointed out the car's front windshield. All audio and video to support this was provided through standard Microsoft Windows NetMeeting being transmitted in IP packets over the ad hoc network. A microphone, audio headset, and video camera were located in the moving car, and a microphone and speakers were located in the hotel; we did not provide video from the hotel to the car since the driver of the car could not watch the video while driving. The video camera used was a Logitech QuickCam Pro camera with a USB computer interface.

In addition to the NetMeeting audio and video data traffic on the ad hoc network, we generated additional background traffic by using one of the stationary cars to "ping" each of the other cars; the source node of the ping was changed every 5–10 minutes, and within this time, each other car was used as a destination of the ping, dividing the time approximately equally between the other cars for this source node.

In most operation of the demonstration, the routes over the ad hoc network ranged up to 3 hops in length. Until the moving car reached the halfway point around the hotel, DSR generally discovered the route of 1, 2, or 3 hops around the "top" side of the hotel, as shown in Figure 2; as the moving car continued around the hotel, DSR generally discovered the route of 3, 2, or 1 hops around the "bottom" side of the hotel. At one point during the demonstration, the laptop in one stationary car was shut down and not restarted, forcing DSR to discover routes up to 5 hops in length in order to route all the way around the hotel building past this disabled intermediate node.

Throughout all runs of the demonstration with our multimedia extensions enabled in DSR, the audio quality received through the speaker was excellent, and sounded roughly the same as for a NetMeeting connection over a standard 56 kbps modem not using DSR. The video quality displayed was also very good, although at times not quite of the same quality as the audio. These observations of the audio and the video quality in

general seemed not to be affected by the frequent routing changes in the multihop wireless ad hoc network as the moving car endpoint node circled the hotel building.

The one factor that did seem to affect the perceived video quality while operating with our multimedia extensions enabled was the behavior of the video camera and NetMeeting as the moving car node turned a corner as it drove around the building. At such times, the rate of change in the video scene captured by the camera that must be encoded by NetMeeting was quite large relative to the behavior when simply driving roughly straight forward. The data rate required by the video during these times thus increased [26]; since NetMeeting gives priority for use of a fixed portion of the available bandwidth for audio data, using only what bandwidth is left available for video, the video image degraded during these times, whereas the audio seemed unaffected.

When we disabled our multimedia extensions in testing during the demonstration, however, the audio and video qualities both suffered significantly. The audio appeared to drop or substantially delay some packets, creating gaps of silence in the audio played through the speakers. During these tests, the video image on the screen also became "blocky" and inconsistent. This is due to the video encoding used by NetMeeting: NetMeeting transmits a full video frame every 15 seconds, but otherwise sends only deltas between these full frames [26]. If a packet is dropped leading to the loss of a full frame, the deltas until the next full frame could not be displayed correctly. After this time, the video image on the screen returned to normal, but after a short while, the degradation would reoccur. Once we re-enabled the multimedia extensions in DSR, both audio and video quality quickly returned to normal and maintained their performance.

## V. RELATED WORK

After our design and implementation of this demonstration, Goff et al independently developed a similar form of preemptive Route Maintenance and of using SNR in Route Discovery [8]. However, they have not implemented their techniques in any real ad hoc network, and instead evaluated them only through simulation. In addition, they based their techniques only on received signal strength, not on signal-to-noise ratio (SNR).

For preemptive Route Maintenance, they used a sequence of explicit request/response packets (which they refer to as "ping" and "pong" packets) to probe the quality of a link after receiving a packet below the signal strength threshold. Although this link probing can avoid some cases in which the link only falls below the threshold for a short time (e.g., due to transient channel fading), it also increases network overhead at the point of detecting the impending link failure, and adds latency before a new Route Discovery can then be initiated if needed. We believe the best approach may be an adaptive probing mechanism that varies the use of probing based on channel measurements.

Also since our demonstration, we have studied the design and use of per-hop flow state maintenance in more detail [10], but like Goff et al's work, we evaluated it in that work only through simulation, not through implementation in a real ad hoc network. In addition, our simulation work there studied mainly only the use of this technique for reducing routing overhead in packet headers, not specifically for QoS support.

Many researchers have proposed schemes for QoS on IP networks, including Integrated Services [3], RSVP [4], RSVP over LSP [2], Multi-Protocol Label Switching [30], and Differentiated Services [27]. Some QoS routing protocols also have been designed for ad hoc networks [6, 7, 9, 11, 20, 21, 31, 32]. These protocols use relatively heavyweight mechanisms to achieve general QoS, and tend to somewhat increase the amount

of routing traffic and overhead. In this work, we instead use lightweight mechanisms based on slight modifications to the existing behavior of DSR to achieve our more specific goals.

## VI. CONCLUSION

In this paper, we have described the design and demonstration of a set of routing protocol mechanisms that substantially improve the transmission of real-time multimedia streams over a multihop wireless mobile ad hoc network. These mechanisms were implemented in the Dynamic Source Routing protocol (DSR) [14, 16], and could also be applied to other on-demand ad hoc network routing protocols such as AODV [28]. We described the implementation of these mechanisms and their public demonstration carrying live audio and video streams over a DSR ad hoc network at the final DARPA Global Mobile Information Systems (GloMo) Principal Investigators Meeting in July 2000. The demonstration consisted of an ad hoc network of 8 nodes, with one node in a moving car driving continuously for periods of several hours or more around the hotel building where the demonstration was held; the endpoint node in the moving car operated a Microsoft Windows NetMeeting session over the multihop ad hoc network to another endpoint node located in the hotel, showing the view from a video camera pointed out the front window of the car, and allowing observers in the hotel to converse interactively with the driver of the car.

Although a number of general mechanisms for providing Quality of Service (QoS) support in ad hoc networks have been proposed, these mechanisms introduce complexity into the system and generally increase system overhead. In addition, published reports of testbed implementations or demonstrations of ad hoc networks [1, 23] do not support these QoS mechanisms. Our goal in this work was to develop a set of lightweight extensions for DSR that perform well and that preserve the basic operation of the DSR protocol, and to gain actual experience with the effect of these mechanisms in a real ad hoc network implementation. The resulting audio and video performance were both very good to excellent, limited mainly by the performance of the video camera and NetMeeting rather than by the frequent routing changes taking place while the moving car continued to drive around the hotel building. In addition, all application and operating system software on the two endpoint nodes in our demonstration were the standard, unmodified commercial version, and all communication was through standard IP packets, thus demonstrating that the DSR protocol and our implementation of it are entirely compatible with this commercial off-the-shelf software.

## REFERENCES

- [1] APE: Ad Hoc Protocol Evaluation Testbed. <http://apetestbed.sourceforge.net/>.
- [2] Daniel O. Awduche, Lou Berger, Der-Hwa Gan, Tony Li, Vijay Srinivasan, and George Swallow. RSVP-TE: Extensions to RSVP for LSP Tunnels. Work in progress, draft-ietf-mpls-lsp-tunnel-09.txt, August 2001.
- [3] Bob Braden, David Clark, and Scott Shenker. Integrated Services in the Internet Architecture: an Overview. RFC 1633, June 1994.
- [4] Bob Braden, Lixia Zhang, Steve Berson, Shai Herzog, and Sugih Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC 2205, September 1997.
- [5] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta G. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '98)*, pages 85–97, October 1998.
- [6] D.H. Cansever, A.M. Michelson, and A.H. Levesque. Quality of Service Support in Mobile Ad-Hoc IP Networks. In *Military Communications Conference Proceedings, 1999 (MILCOM 1999)*, pages 30–34, October 1999.
- [7] Shigang Chen and K. Nahrstedt. Distributed Quality-of-Service Routing in Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1488–1505, August 1999.
- [8] Tom Goff, Nael B. Abu-Ghazaleh, Dhananjay S. Phatak, and Ridvan Kahvecioglu. Preemptive Routing in Ad Hoc Networks. In *Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking*, pages 43–52, July 2001.
- [9] Ying-Kwei Ho and Ru-Sheng Liu. On-Demand QoS-Based Routing Protocol for Ad Hoc Mobile Wireless Networks. In *Proceedings of the Fifth IEEE Symposium on Computers and Communications, 2000 (ISCC 2000)*, pages 560–565, July 2000.
- [10] Yih-Chun Hu and David B. Johnson. Implicit Source Routing in On-Demand Ad Hoc Network Routing. In *Proceedings of the Second Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, pages 1–10, October 2001.
- [11] P'ei hung Chuang, Hsiao kuang Wu, and Ming kuang Liao. Dynamic QoS Allocation for Multimedia Ad Hoc Wireless Networks. In *Proceedings of the Eighth International Conference on Computer Communications and Networks*, pages 480–485, October 1999.
- [12] IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-1997. The Institute of Electrical and Electronics Engineers, New York, New York, 1997.
- [13] Per Johansson, Tony Larsson, Nicklas Hedman, Bartosz Mielczarek, and Mikael Degermark. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '99)*, pages 195–206, August 1999.
- [14] David B. Johnson. Routing in Ad Hoc Networks of Mobile Hosts. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94)*, pages 158–163, December 1994.
- [15] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.
- [16] David B. Johnson, David A. Maltz, and Josh Broch. The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks. In *Ad Hoc Networking*, edited by Charles E. Perkins, chapter 5, pages 139–172. Addison-Wesley, 2001.
- [17] David B. Johnson, David A. Maltz, Yih-Chun Hu, and Jorjeta G. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet-Draft, draft-ietf-manet-dsr-06.txt, November 2001. Work in progress.
- [18] John Jubin and Janet D. Tornow. The DARPA Packet Radio Network Protocols. *Proceedings of the IEEE*, 75(1):21–32, January 1987.
- [19] Ad Kamerman and Leo Monteban. WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band. *Bell Labs Technical Journal*, pages 118–133, Summer 1997.
- [20] Seoung-Bum Lee and Andrew T. Campbell. INSIGNIA: In-Band Signaling Support for QoS in Mobile Ad Hoc Networks. In *Proceedings of the 5th International Workshop on Mobile Multimedia Communications (MoMuC 98)*, October 1998.
- [21] Chunhung Richard Lin and Jain-Shing Liu. QoS Routing in Ad Hoc Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1426–1438, August 1999.
- [22] David A. Maltz, Josh Broch, Jorjeta Jetcheva, and David B. Johnson. The Effects of On-Demand Behavior in Routing Protocols for Multi-Hop Wireless Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1439–1453, August 1999.
- [23] David A. Maltz, Josh Broch, and David B. Johnson. Quantitative Lessons From a Full-Scale Multi-Hop Wireless Ad Hoc Network Testbed. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, September 2000.
- [24] David A. Maltz, Josh Broch, and David B. Johnson. Quantitative Lessons From a Full-Scale Multi-Hop Wireless Ad Hoc Network Testbed. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, pages 992–997, September 2000.
- [25] Marshall Kirk McKusick, Keith Bostic, Michael J. Karels, and John S. Quarterman. *The Design and Implementation of the 4.4BSD Operating System*. Addison-Wesley, Reading, Massachusetts, 1996.
- [26] Microsoft Windows NetMeeting 3.0 Resource Kit. Available at <http://www.microsoft.com/windows/NetMeeting/Corp/reskit/>.
- [27] Kathleen Nichols, Steven Blake, Fred Baker, and David L. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474, December 1998.
- [28] Charles E. Perkins and Elizabeth M. Royer. Ad-Hoc On-Demand Distance Vector Routing. In *Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99)*, pages 90–100, February 1999.
- [29] Joyce K. Reynolds and Jon Postel. Assigned Numbers. RFC 1700, October 1994.
- [30] E. Rosen and Y. Rekhter. BGP/MPLS VPNs. RFC 2547, March 1999.
- [31] R. Sivakumar, P. Sinha, and V. Bharghavan. CEDAR: A Core-Extraction Distributed Ad Hoc Routing Algorithm. *IEEE Journal on Selected Areas in Communications*, 17(8):1454–1465, August 1999.
- [32] Hannan Xiao, W.K.G. Seah, A. Lo, and K.C. Chua. A Flexible Quality of Service Model for Mobile Ad-Hoc Networks. In *IEEE 51st Vehicular Technology Conference Proceedings, 2000 (VTC 2000-Spring)*, pages vol.1 445–449, May 2000.