Test Bed for a Wireless Network on Small UAVs

Timothy X Brown[*], Sheetalkumar Doshi[†], Sushant Jadhav[‡], Jesse Himmelstein[§].
University of Colorado, Boulder, CO 80303

Small (10kg) UAVs are low-cost low-risk candidates for emerging UAV applications. Examples include multi-UAV swarming, flocking, and sensing operations; or, as a communication relay for a network of ground radios mounted at fixed sites, on vehicles, or in sensors. A key enabler is the ability of the UAVs to communicate with each other and with ground based radios as a distributed peer-to-peer ad hoc network. Such networks allow any two radio nodes to communicate directly or through an arbitrary number of intermediate nodes which act as relays. Thus, understanding the performance of such networks in these UAV scenarios is necessary to understand the limits of multi-UAV operations. The University of Colorado has developed and built a wireless network test bed using IEEE 802.11b (WiFi) radio equipment mounted on small low-cost UAVs. This paper describes the testbed and its monitoring architecture. The testbed gives detailed data on network throughput, delay, range, and connectivity under different operating regimes. These results enable us to better document and characterize real ad hoc network behavior among UAVs.

I. Introduction

Peer-to-peer ad hoc (aka mesh) wireless networks of small (10Kg) UAVs will be a key technology in multi-UAV, tactical-UAV, and swarming-UAV operations. In an ad hoc network, radio nodes either exchange packets directly or, if they are out of direct communication range, through one or more intermediate nodes that cooperatively relay the packets. Typically the nodes are mobile so that connectivity changes over time. This mobility is managed by specific ad hoc routing protocols designed to work well in dynamic network environments. Much research has investigated ad hoc networks,[9] including applications to large UAVs.[4] But, the bulk of this research has been in theory and simulation with some limited attempts at real implementations.[8,10] These real implementations often differ from the simulations and theory.[3,5] There is a significant gap in careful studies of ad hoc network behavior, especially in networks consisting of small UAVs. A comprehensive and efficient monitoring scheme can close the gap between real world and simulation. We need a scheme that comprehensively logs traffic information along with the node specific information so that every minute event that occurs in the ad hoc network can be accurately reproduced. The University of Colorado has developed a wireless network test bed using 802.11b radio equipment mounted on small low-cost UAVs. The test bed is located at a federal facility located near Boulder, Colorado. It is a flat 7 km^2 open area suitable for radio and UAV flight operations. More details on the site and testing at the site can be found in Ref. 1. This paper focuses on the monitoring architecture. This architecture will enable us to better document and characterize real ad hoc network behavior among UAVs.

II. Monitoring Approach

The monitoring must reach several goals in order to be effective. The monitoring must provide sufficiently complete information to analyze network behavior in detail. The test bed data should be available in real time. The test bed should scale to 10's of monitored nodes. The monitoring should have minimal impact on the normal operation of the network. In reaching these goals, the monitoring must solve several challenges. The UAV radio nodes are small size, with limited power and payload. The ad hoc networking is complex with control distributed across the ad hoc nodes. Nodes may be disconnected for long periods of time during experimentation and the monitoring should be reliable to these disconnects.

---

[*] Professor, Electrical and Computer Engineering and Interdisciplinary Telecommunications, CB 530
[†] Student, Electrical and Computer Engineering, CB 425
[‡] Student, Interdisciplinary Telecommunications, CB 530
[§] Researcher, Interdisciplinary Telecommunications, CB 530

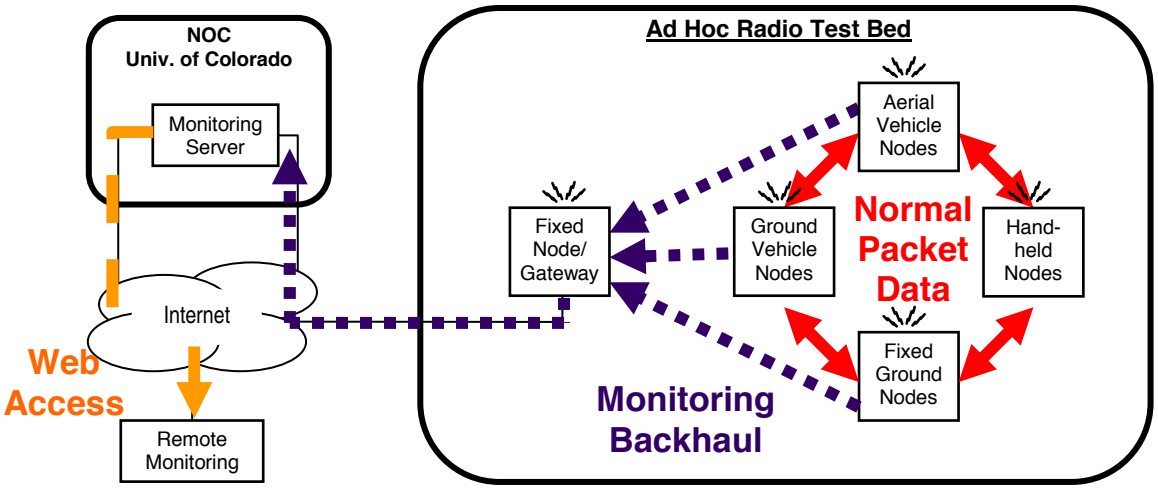

Figure 1. Normal data traffic (red solid) is monitored by each node. Periodically each node sends a report on the data (blue dotted) to the monitor server. This data can be viewed remotely over the Internet (yellow dashed) via a web-based GUI.

These constraints limit some approaches. The real time collection requirement precludes simply storing monitoring data on each node to be collected after the experiment. The limited UAV payload forces the use of the ad hoc network itself for collecting monitoring data rather than a separate radio for monitoring feed back. The distributed behavior suggests that data has to be centrally collected and correlated between nodes. The scaling and interference constraints mean that the monitoring should use minimal computing, storage, and bandwidth resources.

The monitoring approach is shown in Figure 1. The pieces are described in the following sections.

A. Radio Nodes and Gateway

The monitoring starts with the mesh network radio (MNR). The MNR is a compact package that consists of a Soekris single board computer, Orinoco 802.11b PCMCIA card, a Fidelity-Comtech bidirectional amplifier with up to 1W output, and a GPS. The MNR runs the dynamic source routing protocol (DSR)[6] communicating with other nodes via 802.11b. The node can be installed in UAVs and in environmental enclosures for fixed and vehicle mountings.

We implemented DSR using the Click modular router.[2,7] This gives us great flexibility to monitor the network behavior. Running on each node is a monitoring process inserted into the radio packet processing as shown in Figure 2. The monitor collects data on every packet that is sent or received from the node. The information collected is routing level information which includes whether the packet is transmitted or received, the packet type (UDP/TCP/ICMP), packet route (available from the DSR header), time stamp, packet size, and packet sequence number (inserted by the packet source). Higher level application information from the body of the packet is not collected to minimize processing. Lower level 802.11 MAC and physical layer information is not available.

The monitoring also collects information at one second intervals about UTC time, latitude, longitude and altitude of the node's current position from the GPS attached to the node's serial port. A unique feature of the monitoring is an interface for text messages to be time stamped and inserted to annotate the data during operation of the network. Through this mechanism, test scripts running on the node can send messages to the monitoring module so that packet-level behavior can be correlated with the higher level scripts.

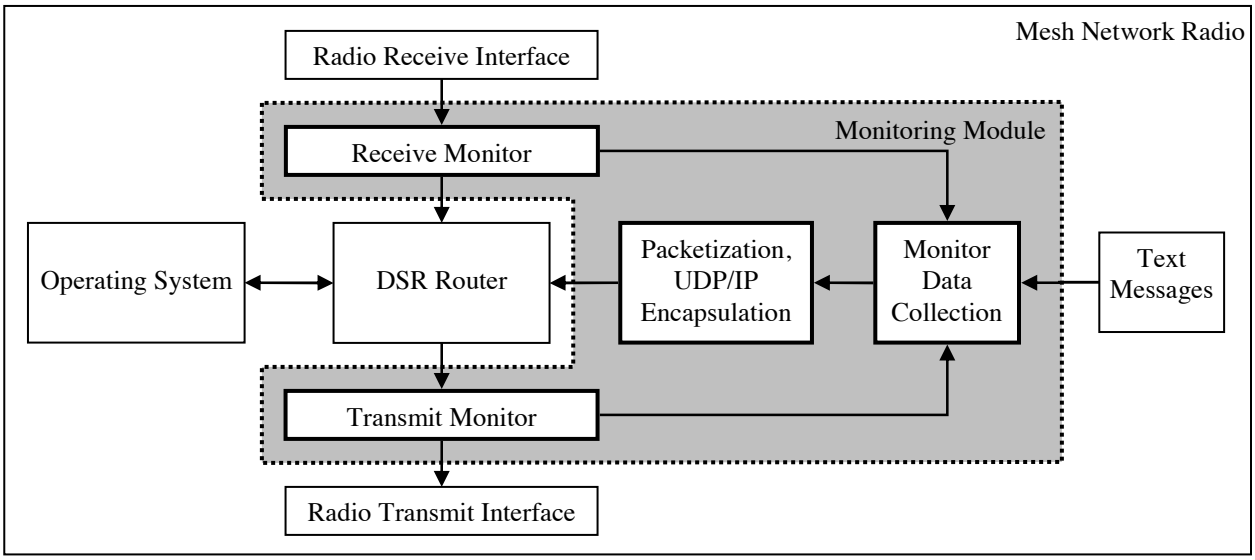

Figure 2. The monitor software (shaded) collects per packet data as packets pass in and out of the radio. This is periodically packetized and sent back to the monitor server.

The information collected by the monitoring module is packetized and a monitor sequence number is added to the packet which is unique per node. Packetization is triggered every 10 seconds or whenever the estimated packet size of the monitoring information equals 1000 bytes. This packet is sent to the test bed gateway.

The gateway may not always be available as nodes move and connectivity changes. Short (few second) outages can be handled reliably by standard network protocols such as TCP. But, a node on the test bed can experience outages of many minutes. For example some experiments may intentionally disconnect the network into one or more subgroups. The subgroups are communicating locally (i.e. there is something interesting to monitor) but may have no connectivity to the gateway.

To handle these outages, we developed a reliable monitor packet delivery mechanism. The monitoring module buffers each monitoring packet and a packet copy is passed on as an application layer packet to a module that adds to it a UDP/IP header with its destination as the gateway. This packet now is passed to the DSR router as a UDP/IP data packet and the DSR router routes this packet to the gateway node. The gateway node DSR router receives the DSR source routed monitoring packet, strips off the DSR header and recognizes the packet as a monitoring packet. It then sends back a Monitoring ACK packet to the node that sourced the monitoring packet. The node on receiving the ACK, removes the corresponding monitoring packet from its buffer and is clear to transmit the next monitoring packet it has lined up in the buffer. If the node does not receive the ACK packet, it keeps retransmitting the same monitoring packet until it eventually gets an ACK from the gateway for that packet. Each retry occurs every 10 seconds. If the packet is buffered for over 1 hour, the packet is dropped and the next packet in the buffer is passed on for transmission.

The gateway strips off the UDP/IP header of the received monitoring packet and adds a new UDP/IP header with the destination as the CU monitoring server. It then forwards on the packet over a wired interface which routes the packet to the CU monitoring server through the Internet.

B. Monitoring Server

The monitoring server receives the monitoring packets from the gateway, parses them, and inserts them into a database. The database is both a data archive and an analysis tool. The database stores three types of data, per-packet data, per-node data, and application messages. These three elements are each related to monitoring data to retain monitoring packet sequence numbers and timestamps on both packet creation and reception at the database.

The per-packet data is the packet data recorded at each node. Note that a single data packet will appear several times in the database since it is transmitted and received by different nodes on its path across the network. Each entry is associated with the point on the path where it was recorded. This level of detail enables a packet to be tracked as it crosses the network and either its successful delivery at the destination or the point where it was lost can be determined.

The per-node data is the GPS time and position data included in the monitor packet. The position of every node at every time during the experiment can be determined. In turn, the distance between any two nodes at any time can

be determined. When combined with the per packet data, it allows packet losses to be correlated with node separations.

The application messages contain both free text and a numeric type to ease sorting and display. Examples include the start and stop times for experiments, the results of the experiments, and notification messages such as when a node powers up or the radio interface is turned off. By embedding this information in the database, the database becomes the complete archived repository of all test bed activities.

This information is stored in a central ODBC-compliant relational database. We are currently using MySQL[**] version 1.4.3 as our database engine since it is open source, freely available, and can be ported to many different platforms. The engine is hosted on a dual-processor 450 MHz Sun machine running SunOS release 5.8. The relational database enables complex queries for detailed network performance analysis. For instance in an experiment to see the effect of a UAV on ground node communications, all the packets can be classified into "not delivered", "delivered via a UAV", "delivered without a UAV".

C. Remote Monitoring Graphical User Interface

The monitoring design also includes real time remote access and data visualization via a Web-based Graphical user interface (GUI). A screen shot of the interface is shown in Figure 3. The GUI is a Java applet (using version 1.4.2) using Sun's standard GUI library Swing to display and analyze network state and performance both post-test and real time. The GUI is divided into 4 logical sections. The Situation Panel, in the upper left, shows the positions of nodes and the routes being used by packets in that time frame. The Options Panel, in the lower left, gives the user control over GUI behavior and test choice, and provides a text box for status messages. The Graph Panel, on the right, displays the user's graphs and provides a toolbar for graph creation. Finally, the Time Bar, at the top, allows the user to maneuver through test results using CD-player style controls, text input boxes, a horizontal slider, and drop down box of playback speeds.

One of the most useful features of the Situation Panel is its support of pluggable maps. Once a map is converted into a supported image format and its GPS coordinates are specified, the user can switch to it very quickly. This allows for maps covering various distances, locations, and resolutions. Maps could also display or highlight different geographical and logical features. The Situation Panel can auto-track and auto-zoom to keep selected nodes in view despite their mobility. The displayed routes can be filtered and can also provide approximate distances between nodes. The GPS coordinates of the mouse pointer are also displayed.

The Graph Panel is designed to allow flexibility and independence of graphs, while providing shared mechanism for time traversal and graph creation/destruction. All graphs share the same x-axis, and therefore the same time frame length and current position. This horizontal alignment of the graphs facilitates graph comparison. The two types of graphs currently implemented are a Message Graph and a Packet Graph. The Message Graph plots the timings and contents of messages. If a message is too long to be displayed in the graph, the user can give it its own pop-up window, or copy it to the status pane. A Packet graph can potentially display any statistic about packets, and currently counts of packets received and sent, both for single nodes and for the links between two nodes, can be created. These data points can be filtered by the routes that packets take and by the types of packets being transmitted. An example of the route filter panel is shown in Figure 4. It shows how packets can be filtered by source, relay, and destination nodes. Another similar panel allows the data to be filtered by packet type (TCP receive, TCP transmit, UDP receive, etc.).

The GUI serves several purposes; experimentation support, data dissemination, and data analysis. The experiments take place over a large area and situational awareness is limited. The GUI enables experimenters at the test bed site to observe node and traffic activity. For instance, when a radio and its GPS are properly functioning, they appear on a situational map in the GUI. Traffic and routing can be monitored during experiments for anomalies. By making the GUI Web-based, the data can be readily viewed by other observers and researchers. Finally, an ad hoc network has many simultaneous activities. The GUI provides a tool for comprehending the big picture and isolating specific events.

[**] www.mysql.com

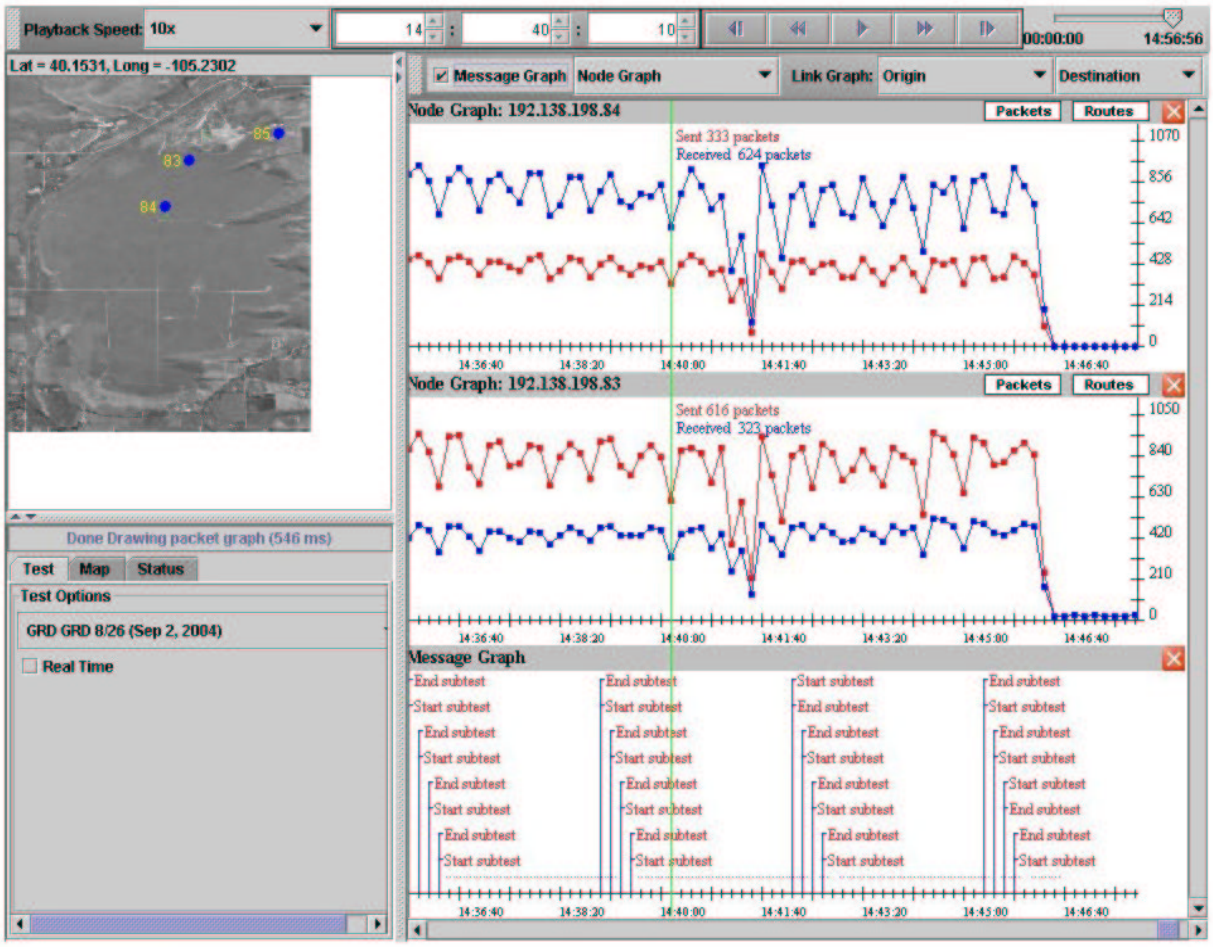

Figure 3. Screenshot from the remote monitoring GUI. Situation map is on the top left showing MNR locations at Table Mountain site. Status messages and control panel are on the bottom left. Performance and message graphs are shown on right. Time control is at the top.

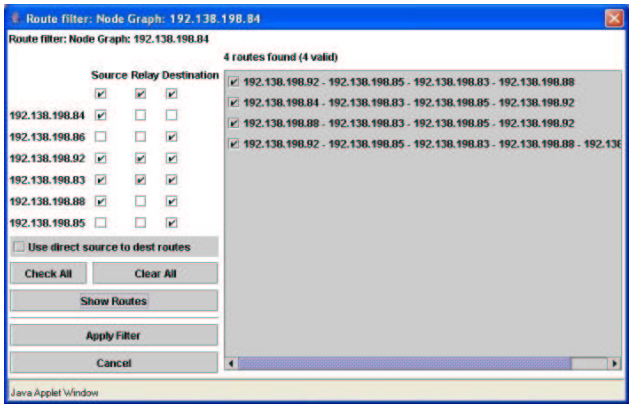

Figure 4. GUI Route Filter.

III. Performance

Monitoring performance was measured along several dimensions. This section describes the results of these performance measures.

A. Network Impact

The first measure is the effect of the monitoring on network performance. For this test the utility netperf[††] was used to measure TCP throughput of a one-hop (direct) and two-hop (one intermediate relay) path. These tests are demanding since they generate a high rate of data packets and acknowledgements that quickly fill up monitoring packets. The rate of monitoring packets in this case is about two per second. The throughput was measured with and without the monitoring and repeated 59 times. The average throughput along with the sample standard deviation is shown in Table 1. The data shows that monitoring reduces the throughput by 20-50kbps. This represents less than 10% of an impact on throughput.

Table 1. The effect of the monitoring software on throughput.

	Throughput (Mbps)	
	1 hop	2 hop
Without monitoring	1.40 ±0.02	0.53 ±0.12
With monitoring	1.38 ±0.03	0.48 ±0.17

The factor of 2.7 drop in performance going from one to two hops is typical of ad hoc networks and represents the double load placed on an intermediate relay node which must both receive and transmit every packet on the two hop path. At this point in our research, these are the kinds of effects that we are trying to observe and a 10% overhead for monitoring is acceptable. Future work is looking at compression techniques to reduce the size and frequency of the monitoring packets.

B. Reliable Backhaul

The monitoring module uses a reliable backhaul to deliver packets to the database. To see how effective this reliable protocol is, we counted the number of packets sent by nodes and the number that did not reach the database. Missing monitor packets are indicates by missing monitor packet sequence numbers. On a typical series of tests, 15,528 monitor packets were generates and only 36 were lost (0.23%). Given the irregular connectivity available, this 99.8% reception rate is good. Further analysis revealed that these losses occur from double acknowledgements being generated. This has been fixed and monitor losses in future experiments will be less. To further see how effective this method is we plot the cumulative density function (CDF) of the delay from when a monitoring packet is generated to when it is delivered to the database in Figure 5. Half of the packets are delivered within a second, but, 5% of the packets required more than a minute and 0.1% required more than 10 minutes. Thus, the need for a dedicated reliable delivery mechanism is clear.

Looking carefully at Figure 5 shows that the CDF curve rises in a series of 10 second steps. These steps correspond to the 10 second retransmit times of the reliable delivery mechanism. Assuming that a monitoring packet is delivered within 10 seconds on a given retry if the retry is ever to be successful, then these steps indicate the number of packets that are successful per retry. The first try delivers 80% of the packets in the first 10 second interval. The second try only delivers 30% of the remaining packets. The third try delivers only 22% of the remaining packets and so on. This approaches a 20% asymptotic delivery rate per try. These numbers suggest that most nodes have a reliable connection to the gateway, but, a few nodes have a persistent low probability of connect again supporting the need for the reliable delivery mechanism.

[††] www.netperf.org

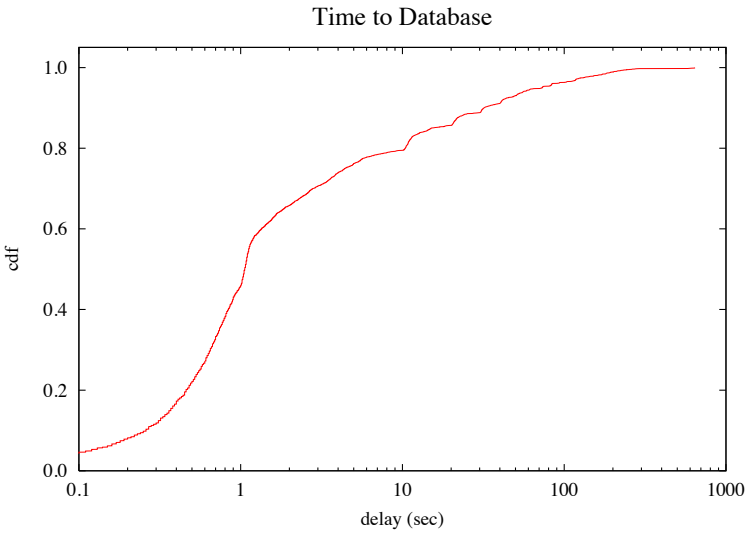

Figure 5. Probability Monitor Packet is Delivered within Specified Delay.

C. Database and GUI

Several aspects of the database and GUI were measured. For an example test of 6 nodes for 25 minutes, the database stores 4.8 MB of data unique to the test. 54% of the data handles direct packet-level information, and 46% is monitoring and relational information used for linking to the rest of the database.

The code to insert into the database has been profiled and tuned to be quick as possible while still using the SQL interface. By inserting data in blocks and creating our own unique keys, we can insert 2742 monitor packets, 25 minutes worth of data for 6 nodes, in 14 minutes. The current database server is a 450MHz processor machine. A machine with a faster processor would allow proportionally faster inserts.

The GUI loads all available test data for a given test at the moment it is selected. For a sample test of 6 nodes over 25 minutes, this takes 24.5 seconds, or close to a second of loading time per minute of data. As the data is loaded, the GUI creates an internal representation of the test data as collections of Java objects. These objects are filtered and aggregated to produce information suitable for display. The GUI performance is optimized for "play" mode and displays for any time increment at a rate of one increment per second.

IV. Conclusion

The monitoring has four goals; sufficiently complete, real-time data presentation, scales to 10's of nodes, and minimal impact on network operation. The detailed per packet data combined with the reliable packet delivery satisfies the first goal. The GUI interface provides immediate access to data as it enters the database and it can be presented in real time. Current network testing has used up to 8 nodes. A larger network with several simultaneous file transfers will generate monitor packets at a higher rate. To keep up with the proportionally greater monitoring traffic, the monitor architecture can be migrated to a faster database server. To minimize monitoring traffic in a larger ad hoc network, multiple gateways can be installed to reduce the distance the monitor traffic must traverse before being removed from the network. The current monitoring module reduces each node's performance by less than 10%. This is sufficient for current test bed goals. This impact can be reduced through more compressed monitor packets to reduce their number and size.

The test bed is being used to evaluate the DSR protocol over the 802.11b MAC. The monitoring has no operational dependence on the MAC. The only dependence on DSR is the DSR packet parsing in the monitoring module and an optional sequence number header that was added to the DSR router. Future work will address other ad hoc routing and MAC protocols.

Acknowledgments

This work is supported through a grant from the L3-Comcept Corporation. Thanks to the Institute for Telecommunication Sciences for providing the Table Mountain site. Thanks to Gerald Jones for initial support of the GUI interface. Thanks to Brian Argrow, Corey Dixon, Jack Elston, Jake Nelson, Philip Nies, and Bill Pisano for

their part in the design, construction, and operation of the UAVs. Thanks to Daniel Henkel, Marc Kessler, and Roshan-George Thekkekunnel for testing support.

References

[1] Brown, T. X, Argrow, B., Dixon, C.R., Doshi, S., Thekkekunel, R.G., Henkel D., "Ad Hoc UAV Ground Network (AUGNet)," *AIAA 3rd "Unmanned Unlimited" Technical Conference*, Chicago, IL, 20-23 Sep 2004

[2] Doshi, S. Bhandare, S., Brown, T. X , "An On-demand minimum energy routing protocol for a wireless ad hoc network," *Mobile Computing and Communications Review*, vol. 6, no. 2, July 2002.

[3] Feeney, L., Nilsson, M., "Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment," *IEEE INFOCOM 2001*

[4] Gu, D.L., Pei, G., Ly, H., Gerla, M., Zhang, B., Hong, X., UAV Aided "Intelligent Routing for Ad-Hoc Wireless Network in Single-Area Theater," *Wireless Communications and Networking Conference (WCNC)*, IEEE , Volume: 3, 23-28 Sept. 2000 Page(s): 1220 -1225 vol.3

[5] Heusse, M.; Rousseau, F.; Berger-Sabbatel, G.; Duda, A.; "Performance anomaly of 802.11b," *INFOCOM 2003*. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE, Volume: 2, 30 March - 3 April 2003 Page(s): 836-843

[6] Johnson, D., Maltz, D., "Dynamic Source Routing in Ad HocWireless Networks," *Mobile Computing*, Chapter 5, pp. 153-181, Kluwer Academic Publishers, 1996.

[7] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems*, vol. 18, no. 3, pp. 263–297, August 2000. http://www.pdos.lcs.mit.edu/click/

[8] Maltz, D., Broch, J., Johnson, D., "Experiences Designing and Building a Multi-Hop Wireless Ad Hoc Network Testbed," *CMU School of Computer Science Technical Report CMU-CS-99-116*, March 1999.

[9] Royer, E., Toh, C., "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," *IEEE Personal Communications*, April 1999, pp. 46–55.

[10] Sanghani, S. Brown, T. X, Bhandare, S., Doshi, S. ``EWANT: The Emulated Wireless Ad Hoc Network Testbed," *IEEE Wireless Communications and Networking Conference (WCNC)*, 16-20 March, 2003.