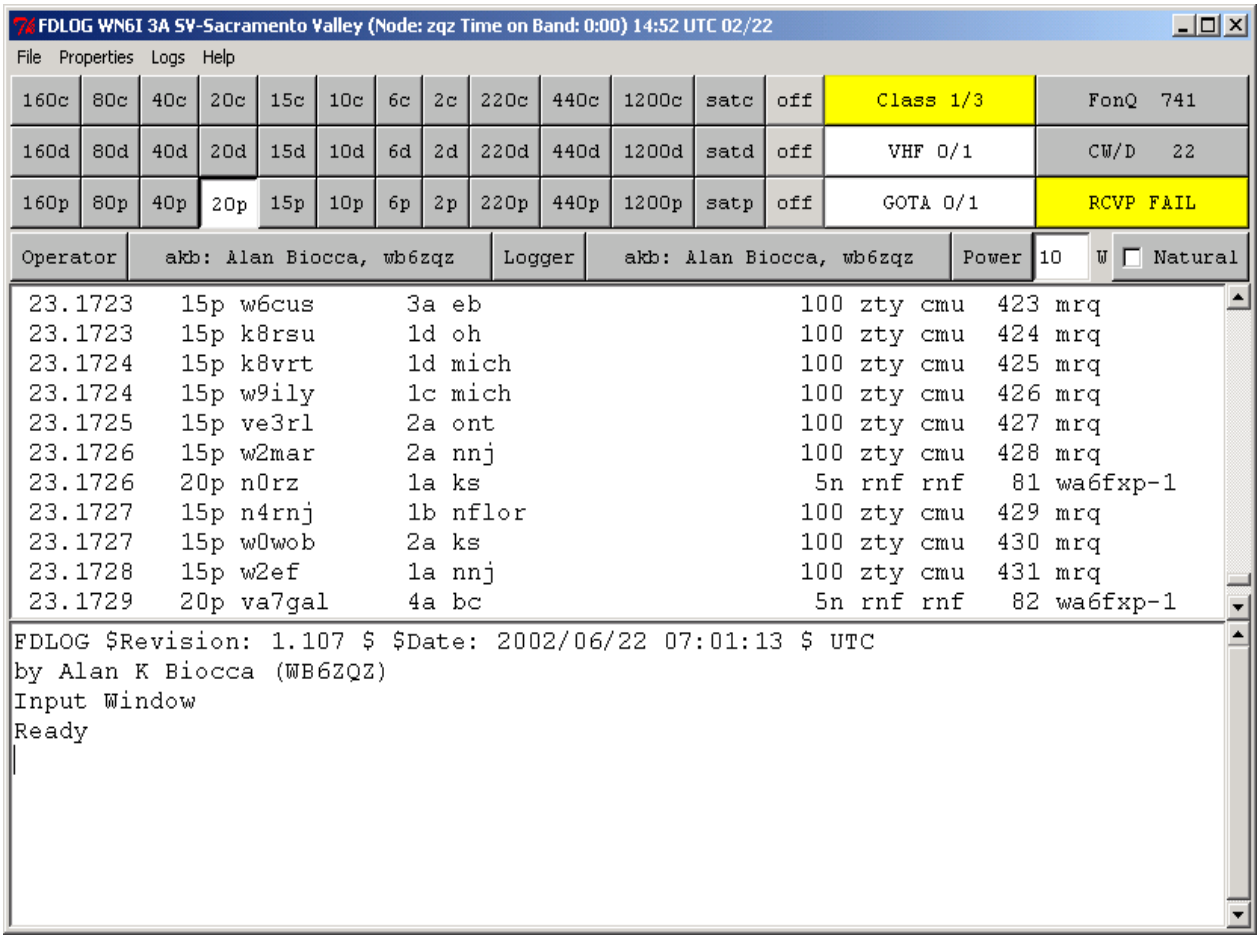
A Reliable Logging System for Field Day based on Wireless Networking

By Alan K. Biocca

6 March 2003

This is the article essentially as published in "CQ VHF Spring 2003 page 18". For further information on the development progress of this software refer to the author's web page "www.qsl.net/wb6zqz" or send him email via wb6zqz at arrl.net.



Example FDLOG Screenshot

Last year on Field Day (FD) 2002 we deployed a new Contact Logging System developed to address the problems presented by coordinating multiple stations and maintaining a log database under field conditions. It is based on low cost IEEE 802.11b wireless networking technology configured using the peer-to-peer (ad hoc) model and free software. The capabilities of the wireless networking facilitated a more convenient and functional system than had been practical before. A synchronized replicated database solution provides high reliability, performance and scalability. This article describes the development of that system and discusses some other useful applications for which these techniques could be applied.

Our Field Day group is a technically oriented group. Our focus is on problem solving, and FD is a problem-rich environment. We put a lot of effort into preparation and planning - selecting equipment, making cables and antennas, and Antenna Launching systems.

Logging contacts during FD with paper is a real chore - especially when checking for duplicate contacts. The old paper dup sheets worked pretty well when there were only a couple of prefixes, but with wide variations in callsigns it is no longer so simple to handle with paper. Logging by computer was once somewhat of a luxury but is now is a necessity.

There are lots of computer programs available for logging, and they have scores of features, but I have not found any that meet our requirements quite as well as I would like. My systems have focused on meeting the fundamental requirements well, and not provided a lot of extra features. One of my goals is to make the Field Day event more enjoyable, and to make it very efficient for the people involved and logging software is an important component of the solution.

Requirements for an Excellent Field Day Log System

The term "station" refers to a set of radio equipment, the term "node" refers to a computer used for logging contacts. These two don't correspond directly - a station may not have a computer logging node, and then its contacts would be entered later, and there may be additional nodes around the site that are used for monitoring and communications purposes that are not associated with a station.

First I will discuss some of the essential requirements, and then the solutions we have developed for the system.

Chris KG6LXL and Sharon N6MWD

Ease of Data Entry

One requirement is to minimize the number of keystrokes that are required to perform the process. The cycle of checking a callsign for a potential duplicate contact, and logging a contact must be quick, efficient and straightforward. Many folks do not type very fast, especially when one person is both operating the radio and logging, thus it is even more important to minimize effort (keystrokes).

Constrain the Mouse

The keyboard is more efficient for data entry than the mouse. Moving the hand between keyboard and mouse is slow and cumbersome, so one requirement is to eliminate the mouse from the actual logging cycle. It is acceptable to use the mouse for other less frequent activities.

Efficient and Effective Station Coordination

Another requirement is to facilitate the coordination of the stations. In our operations the stations are allowed to change bands as they wish, but we must prevent multiple stations from occupying on the same band segment, as well as prevent

exceeding the number of stations we have decided to operate (the Field Day Class). For example, we may have five or six stations set up (each with somewhat differing but overlapping band capabilities), but be operating a two or three transmitter class. We need to know which stations are actually operating at any given time. If we decide to change the Class during the contest it needs to be communicated quickly and effectively to all stations, since it affects the report they are to give for each contact. (Field Day rules permit increasing the class during the contest).

Contact Duplication Detection

Since our group's several Field Day stations may use each band at some point during the operating period, they must all have access to the entire log data set for duplicate contact checking. This means that any station working a band-mode must have all the duplicate log data for that band-mode. (Field Day rules regard band and mode as separate contacts as far as duplicate contacts are concerned - modes are phone, digital and cw).

Time Synchronization

All nodes must have accurate and synchronized time bases for time stamping data. Log entries require timestamps, and they should be accurate and consistent within a second or two across all nodes.

Database Integrity Hardening

Interruptions in processing - whether caused by power failures or system crashes - should not cause database corruption. If corruption does somehow occur it should be possible to recover with a minimum of impact on the overall operation.

Frank WB6MRQ

Minimized System Component Dependency

The system must be highly available and very reliable. It should not matter if a person, or some equipment is suddenly unable to attend the event, or if some equipment fails or must leave the site before the event's conclusion. The system should keep operating as well as practical. The amount of difficulty caused by a single failure should be minimal and temporary. Backups and spares should be available to insure system availability. It should be possible to add a system node (computer) during the contest without significant delay (more than a few minutes) or negative effects on the system as a whole.

The system must be scalable from a single node to at least two dozen nodes. Our Field Day operation is generally small, but some groups field 15 or more stations.

Each system node should function with a minimum of equipment, and the gear should be common, available and relatively inexpensive. At a minimum each station will require one node, plus a couple of spare nodes for the site.

Development

The first system was developed in 1984, based on the equipment and software tools available at the time. The Heathkit H89 computer running CP/M was the best machine I had available to take to field, so I wrote the software to facilitate an effective deployment and to meet many of the requirements. I used the 'Software Toolworks' C compiler by Walt Bilofsky (N6QH) in conjunction with a small multitasking kernel that I had developed previously to facilitate handling (might this not be said better?) multiple simultaneous users within one program.

The power for our Field Day operations was provided by a Generator, and was subject to interruptions, mostly scheduled for refueling. This was not convenient, especially for the computer, so I developed a 40 hour continuous operation capability using an external fuel tank and low pressure electric fuel pump. This handled the routine outages, but the occasional random shutdown was still a possibility, and on a couple of occasions over the years power outages did occur.

The H89 computer was built into an H19 terminal, and could be equipped with a three port serial card that allowed for the addition of two more terminals and a serial printer. The H89 had been upgraded to have two internal floppy drives. This allowed for the protection of the log data by writing it alternately to files on each floppy. This way, if the file was corrupted due to power loss during write, there a recently copy of the file remained on the alternate floppy. Additionally, the incoming log entries were printed to an Epson MX80 printer creating a paper record to fall back on in case of computer problems. Thus, the system was protected against power outages proved by subsequent outages and the fact that we did not lose log data..

Mike WA6ZTY

The user interface was developed to meet the user efficiency requirements, and this minimum keystroke/quick response approach turned out to be very popular with the group. Efficient hash based data structures were implemented so the duplicate checking took only milliseconds. This was a very effective system, but it did not meet all the requirements.

Past Limitations

The H89 system could only support 3 stations, but at the time it was adequate for our operations and performed well for many years. We strung RS232 cabling through the forest, and kept the stations close together so we could reach the central computer. On one occasion a long three-wire power cord extension was temporarily adapted (haywired) for RS232 service. I considered making RS232 to U-ground power cord adapters to facilitate this, but visions of smoking computers prevented me from proceeding on that project.

One problem with the H89 solution was the single point of failure - no one else in the group had duplicate hardware. One year there was a hardware

problem and we were luckily able to replace the bad RS232 driver chip with a spare, but had it been something more critical we might have been back to paper logs. I always carried a pack of paper logs and dup sheets, but we never had to use them for primary logging.

Our FD group is somewhat unusual in that we are not a regular monthly 'club', and we deploy into the local National Forest. The makeup of the group, the location, the number and type of stations are all dynamically determined each year to a larger degree than many FD operations. Further, we don't go to the same spot, and we run different numbers and types of stations most every year. The group changes also - often significantly.

The age of the hardware prompted us to change systems. Eric WD6CMU provided an OS9 operating system based system (68000 processor) and ported the code. (OS9 is an operating system intended for real time systems). This added a hard drive instead of the dual floppies. Eric also set up a makeshift Uninterruptible Power Supply (UPS) with an inverter and a battery to reduce the danger of corrupted files on the hard drive should the generator quit during a write.

In terms of meeting the requirements, this system was substantially similar to the H89 based one in aspects of scalability and single central point of failure - the OS9 computer itself. We used it for a number of years until moving on to an MSDOS based system.

The OS9 hardware was getting old and cranky, so Steve KA6S, Eric WD6CMU, Rich WA6FXP and the author moved the code to MSDOS. The target computers selected were primarily portable MSDOS machines such as the Toshiba T1000 (which were plentiful in the

Eric WD6CMU

group), and this shaped the result. Gone was the central database - instead there were floppies for each band and mode. Now every station had its own computer and there was good redundancy, but getting the right floppy was a problem, time synchronization was poor, and getting all the data into one report after the event was a real chore.

I worked on ways to network the machines, but networking with MSDOS based laptops was not trivial. I collected Apple type Local talk transformers and designed a homemade network that connected to the serial port, but this never made it past the theory and parts gathering stage.

Desktop PCs and Laptops were capable of a lot more than the T1000, but we had to work with what people were willing to take to field. There were Radio Frequency Interference concerns as well with PCs. Some PCs were starting to show up at Field Day, primarily for digital modes, so I prepared to make a network in the forest, procuring a thousand feet of Category 5 network cable and a couple of Ethernet hubs. Network cards were by then inexpensive. But what would happen with RF transmitters right next to Cat5 cabling? Stretching ca-

ble in the forest is possible, but is it practical? Some of our stations get spread out, depending on the trees available for shade and antenna support. Powering the hubs was somewhat of a problem, though a deep cycle battery and regulator could do the job for those models rated below 12 volts.

I developed a new program based on a web server and database model using free software including the excellent PHP web programming language, the Apache web server, and the MySQL database on a Linux host. This had the interesting property that we could test it on the internet, group members logging to the web server from home, which we did.

This system did not meet the efficient keystroke requirements, and based on user feedback was not deployed during Field Day. The effort required to improve it with client-side Javascript or Java programs appeared to be substantial, and it still was not clear that the performance would even then meet the expectations of the group - who was accustomed to a very interactive keystroke by keystroke application. This approach also suffered from the single central point of failure problem. I brought it along on one Field Day to test it, but we couldn't get it to work on Eric WD6CMU's Linux laptop, demonstrating the single point of failure (though we hadn't planned to use it primarily anyway). So we used the T1000's another couple of years.

I kept looking. Wireless networking appeared to be a good replacement for the cable, if we had enough compatible equipment. There were multiple standards, and hardware was expensive. However, upon the commercial success of IEEE 802.11b (hereafter referred to as 802.11b) and its emergence as the wireless network leader, the hardware became low-cost. We deployed it at work and at home, and obtained experience with it. I studied it to understand how the different configurations might work in the field. Some tests were made. Group members had enough laptops and wireless hardware to do this. But what about software? I looked for available software meeting our requirements. What I found did not appear to meet our requirements very well and the old program was not a good candidate for upgrading. It was time to redesign.

For many years I had considered a replicated database approach to the logging problem. In this model each computer maintains a full copy of the database. This makes it easy to do duplicate contact checking on any band, or make the report for the contest entry. It meets the reliability requirement well - any failures of one node in a multi-node replicated system do not lose data. The difficulty is to maintain the replication - i.e., keep the databases the same. A simple way to do this occurred to me a long time ago and is based on the Usenet Newsgroup Article Flood-Fill algorithm. This is an old technique, even predating the web. Each node has periodic exchanges with a few other nodes, and the two conversing nodes determine if either has any 'items' that the other does not. If so, they exchange them and each update their databases. In this manner a new 'item' floods across the databases in fairly short order. I envisioned a system in which the contact information floods across the multiple copies of the database, so every station has the full log for duplicate checking and reporting.

I looked at various languages to implement this project in, and decided that Python was a good choice. I did not yet know the language, but research showed it to be appropriate for this work.

Python supports the required capabilities such as threads and network sockets. Several compatible Graphical User Interface (GUI) libraries are available. Additionally it supports efficient rapid development which is important for me since this is a “spare time" project. In several weeks I had a command-line version of the new FDLOG program working. I selected a GUI library called 'Tkinter' and moved it into the graphical programming model. This is a library that uses the 'Tk' GUI toolkit that was originally developed for the Perl programming language. In a few more weeks the program was ready for alpha testing. The Internet was instrumental in the development, supporting our FD group discussions and distributions of new test software. The development was done on Windows 2000, but Python is platform portable, as is Tkinter, so the resulting program runs on Linux, most Windows versions and even the Macintosh. The Mac version of Tkinter seems to have problems with some of the fonts I used, so it does not look good (in fact it is hard to read some of the buttons), but we did not spend much effort to see if this could be improved. Supporting multiple platforms helps increase the amount of compatible equipment that we can use, so this was a real benefit.

FDLOG Network Protocol Components

The FDLOG program uses the Internet Protocol (IP) based User Datagram Protocol (UDP) rather than the more common Transport Control Protocol (TCP) TCP was designed to handle streams of data reliably for Telnet and File Transfer Protocol (FTP). It was designed to support point-to-point transmission of large volumes of data. Setting up and breaking down a communications channel is somewhat complex and it does not support broadcasting by its point-to-point nature. These characteristics did not meets our requirements. UDP is designed to carry small 'Packages' of data called ‘datagrams’ with low overhead and it does support broadcasting. There is essentially no overhead to set up communications but there are costs.

The cost comes from UDP characteristics and must be addressed. The immediate issues were, with UDP - datagram arrivals are not guaranteed - they may be lost, duplicated (delivered more than once), or delivered out of order. It is up to the application to handle these problems, whereas in TCP the application is guaranteed that the stream of data will arrive correctly and in-order. Of course the TCP connection itself can never be guaranteed (a computer may crash), so the TCP problem is in detecting and re-establishing lost connections, and in many cases this complexity exceeds that of dealing with the UDP issues. TCP also handles modulating the data rate to efficiently fill the network pipelines for large transfers, where as FDLOG uses simple timers and by design only uses a small fraction of the network bandwidth to retain good real-time performance.

The FDLOG program requires broadcasts for two purposes. One is to periodically share the status of a computer's databases and clock with its neighbors. The other is to share a brand-new database entry. By broadcasting new database items they are efficiently distributed to all other nodes that are within range. Out of range nodes discover new items from a broadcast of a neighbor and then request the 'fill' of the missing items - the aforementioned flood-fill algorithm. The request and fill are point-to-point 'directed' UDP packets. A simple timeout and retry mechanism han-

dles lost packets, and duplicate packets are automatically rejected as the data is already in the database. Since the node asks for one fill at a time there aren't really any out-of-order packet issues. Randomness was introduced into the source selection process so that requests are distributed to the various neighbors of the requesting node to avoid loading one machine or getting stuck on a poor path or failing node.

The process worked well in tests at home with two or three machines. It was extremely interesting to bring a new node onto the network. It would listen for a few seconds and upon receiving a status broadcast it would detect that it was missing all the data the other node had, so it would immediately start requesting the missing data in rapid-fire. Soon it would be requesting all the nearby nodes for data at the maximum rate until it had all the data. I chose a rate that would not load the network significantly, but would catch up a completely empty new node in a few minutes, at least for the database sizes expected in Field Day logging system usage.

Wireless Configuration Options

While it was clear that 802.11b wireless networking would be more convenient than Cat5 cable in the forest, there was still the question of how to configure it. Two modes of operation were available - Peer to Peer (called Ad-Hoc in the IEEE 802.11b standard) and Access Point (called Basic Service Set or BSS) configurations. The Access point configuration is easier to set up - one node is set up as the primary 'Access Point', and it uses dynamic host configuration protocol (DHCP) to assign IP numbers to all the client nodes. At the same time it assigns them a number it also issues them the rest of the network parameters. This is the usual configuration for wireless systems in the workplace and at home (and works with clients configured to “obtain an address automatically”), so in most cases merely rebooting the client computers would bring up the wireless network. The configuration complexity is then limited to the Access Point node itself (and most home type wireless equipped routers have adequate default settings). The disadvantage is two-fold. First of all, we would have to bring an Access Point node, or configure a computer to perform that function. Since it would potentially be a single point of failure, we would have to provide duplicate backup hardware. Power would have to be provided for this 'extra' node. The Access Point would also perform as a repeater - which would have to be able to reach all the clients (which could be a problem), and it would repeat all their packets - which reduces the system bandwidth by half.

The other possible configuration, Peer-to-Peer networking, referred to in the 802.11b specification as “Ad-Hoc” mode wireless, has the advantage of no single point of failure in the system. This configuration maintains the full bandwidth capability of the system by not requiring all packets to be repeated by an Access Point, or it increases the effective range by only requiring that each station be able to hear some of the others, and that all such subsets overlap in order for the FDLOG FDNet protocol to perform the data flooding algorithm. Thus, any station can come or go at any time and the system will continue to operate. The disadvantage of the Peer-to-Peer configuration is that each node must be configured manually with network information, and as we found out, not all 802.11b hardware/software combina-

tions inter-operate well. Nevertheless, for Field Day 2002 we selected Peer-to-Peer because it was best suited to fulfill the operational requirements. Note that FDLOG works within the broadcast range of the network, so either configuration, or even wires in the forest, will work.

The first actual group test of FDLOG's networking was performed at the pre-planning meeting for Field Day 2002. We met at Weo WN6I and Sharon N6MWD's house, and after doing the usual FD planning and consuming a few pizzas we retired to the laptops. Folks dispersed throughout the house and we worked on configuring 802.11b to run in peer to peer mode. This turned out to be tricky. Each card manufacturer has different software drivers, and the terminology, settings and capabilities are in some cases considerably different. Some cards did not work correctly in peer to peer mode. Some were fixed by downloading new drivers and firmware from their respective manufacturers. Most folks ended up using the Lucent Gold (also marketed as Orinoco or Avaya) cards which are known to be very high quality. I did get my SMC card to work. The Lucent type cards appeared to have more range than Intersil's Prism chipset based SMC, perhaps because it appears that the Lucent design uses Logarithmic AGC whereas it appears the Prism uses Linear AGC. This observation was based on signal strength measurements using the Lucent software which reports signal readings from both ends of the connection.

More effort went into this configuration process than I expected. I had a mini-web server running on one laptop with the Python software, the FDLOG program, and the NTP client software. So after each user configured their laptop on the peer-to-peer net they could download the latest software via the wireless net. The mini-web server is part of the Python distribution, so it was very convenient and worked well for this purpose. A couple of small issues with the FDLOG software were noted and addressed later. The test was a success and it proved the software and wireless network was viable for Field Day!

Time synchronization was still a concern. Brad N6BDE was working on a way to setup an NTP timeserver. We setup NTP client software (Automachron) on the clients. The FDLOG program does report time errors exceeding an adjustable window, so it would 'warn' us about machines that were way out of time sync. My fallback plan was to walk around with a handheld GPS (Global Positioning System) receiver and supervise setting each clock, if necessary.

Example Screenshot Description

The FDLOG screen capture shows some of the program's features. The title bar shows the group callsign, class and section, the node name and the time on the current band and the present UTC time and date. Below the menus are the band select buttons, this example showing the local node's station active on 20-meter phone. The color coding of the buttons shows the bands of the other stations and the CLASS, VHF, and Get On The Air station (GOTA) displays show the current count and full count of the transmitters for the chosen CLASS at the moment. From these displays an operator can see what bands are in use, if there is a conflict of two stations on one band, if his station is in conflict, and if there are any available transmitters in the class the group has selected. To the right the last column of readouts displays the counts of phone

and cw/digital contacts, and the condition of the network. In this example the network shows an error “RCVP FAIL” because there are no received FDLOG packets since I'm running this standalone against the actual log database from our group's Field Day 2002 for the screen capture. Below that the operator and logger are the operator and logger selection buttons, along with the transmitter power, and a checkbox for natural power. Below the buttons is the scrollable log window which displays all log activity from all stations. The bottom window is the input area where new contacts and commands are typed.

I was in the midst of making some minor improvements to FDLOG when development time ran out and Field Day itself was upon us. With a very late thaw (for California) we just were able to get the group into the 2002 site at the Iron Mountain ski lift area in the El Dorado National Forest. There were patches of snow on the ground, and Weo WN6I's minivan and folding trailer required some towing assistance from Mike WA6ZTY's sport utility vehicle (SUV) to negotiate the last bit of the road into the site.

After setting up camp and completing dinner we settled into some final testing of the software. One bug was found and the culprit - an extra comma - was deleted and after which we appeared to be ready for Field Day 2002 to begin the next day.

Eric WD6CMU's new Compressed Air powered Antenna Launcher was the focus of Saturday morning, and an array of antennas and stations went up efficiently. Mike WA6ZTY's 360 foot per leg Vee beam took awhile, mostly to select the just-right trees for spacing and included angle. Eric skillfully sailed the lines accurately over the tip-tops of the hundred foot trees with height to spare.

Our preliminary testing and preparation paved the way for a smooth and efficient field setup of our wireless network. However, Brad N6BDE struggled with the timeserver even though it had worked fine during testing. But here, at 8500 feet elevation and many miles from home, it refused to lock on to the satellites. He eventually got it to work though in the meantime we used handheld GPS units to set computer clocks.

All in all we had six stations set up, and about eight laptops on the wireless network. Half the group's contacts were made by Mike WA6ZTY on his tremendous Vee beam, and the wirelessly networked FDLOG performed well. All over the site were laptop computer screens showing the band utilization, and the scrolling log of all contacts from all stations. The biggest problem was viewing the LCD displays in the bright sunlight, which occasionally caused operator errors, such as entering contacts on the wrong band. Next year we'll have to make some hoods to improve the display visibility, and perhaps improve the editing functions for fixing batches of incorrect entries.

After Field Day we move on to other things, and preparing the ARRL entry is not the fun part of the activity. Occasionally this has led to missing the due date for entry submission. To help this process along the FDLOG software was set up to prepare the submission. This requires some extra information entry, but once done the entry form is generated into an ASCII text file which can be combined with other material such as pictures, and submitted to ARRL directly via email.

The FDLOG software is available at my website, see the links at the end of this article. Other links are also included to the Python software, the Tkinter

graphics library, the Time Synchronization software and the Tennis Ball Antenna Launchers. The FDLOG software has not received any development effort since last Field Day, but I may make some improvements for this year. Check the website for details.

Other Applications for this Technology

Some years back a program called "ARES Data" was developed by Weo WN6I and Dave N6KL for emergency services resource management. It provides a central database that is accessible from packet radio and supports multiple simultaneous users to share a database keeping track of resources or whatever was needed. As I designed FDLOG it occurred to me that the synchronized database system could perform this class of application without the access performance bottleneck and single point of failure of the one central database approach. Nodes can be networked with multiple peers to increase reliability. A combination of wireless and wired Internet links can be used, even packet radio can be added to the system if bandwidth is sufficient. A fairly free format database with flexible user interface could allow it to be quickly adapted to a range of problems. A powerful search engine could make it a sort of 'private web' for the emergency management. This project has not been undertaken but would be a natural variant of FDLOG.

Field Day Group

Thanks to all the members of our "High Sierra" Field Day Group, for providing feedback and testing the software, and reviewing this article. Currently active members of the FD group include: Frank WB6MRQ, Eric WD6CMU, Rich WA6FXP, Brad N6BDE, Mike WA6ZTY, Weo WN6I, Sharon N6MWD, Daniel KG6CNX, Kit WA6PWW, Ken WB6MLC, Oliver KB6BA, Glenn WB6W, Judy KF6MBH, Cal KA6BOI, Steve KA6S, Dawn KB6LHP, Chris KG6LXL, Alan WB6ZQZ.

Special thanks to Ted Sopher for detailed feedback on IEEE 802.11b issues.

Internet URLs

FDLOG software website - www.qsl.net/wb6zqz/

Python language - www.python.org

Tkinter graphics library - www.python.org/topics/tkinter/

Automachron ntp client - www.oneguycoding.com

Tennis Ball Antenna Launcher - www.qsl.net/wd6cmu/, www.qsl.net/wb6zqz/