

## **The P25Proxy Application for remote messaging**

Kaitlin Wilson-Remmer

### **Introduction**

As part of the Project54 system, information is conveyed from the cruiser to a server at headquarters (HQ) via digital radio. Information about each cruiser, such as Project54 software versions currently running, that is sent to HQ is kept in text files at the server as a method of record keeping. In order to take full advantage of the Radio Control Protocol (RCP) [1] feedback from the Motorola radio, a proxy application was integrated into the P54 system to handle radio data communications. This proxy, called P25Proxy, was named for Project 25 (P25) which helped to standardize two-way radio communication so that in an emergency situation, all two-way radios of responding departments are compatible ([www.project25.org](http://www.project25.org)). P25Proxy removes much of the burden from the individual applications that send messages to the server, thus creating an easier method of development in the future. In conjunction with this change, the structure of the server that handles this communication was converted to the Project54 style. The result is a server that contains an Application Manager, proxy and individual applications and the entire client-server conversation can be performed using Project54 Version 2 Messaging [2].

## P25Proxy structure

The underlying structure of the P25Proxy is taken directly from the current proxy employed for handheld devices [3]. The objective of these proxies is to render the underlying communication network invisible to components within the Project54 system. This is accomplished through aliasing, where the proxy is known to the application manager not only as its own name, but also as the names of remote applications.

Version 2 messaging follows the format: Message(source,destination,id,message\_text) [4]. For an application to communicate to the P54-style server, it only needs to know the name of the application it would like to send a message to. For instance, the application at the server that handles incoming GPS reports from the cars is currently known as: avlservice. For the GPS application in a car to send a message to this server application the first two fields would be filled as: Message(gps,avlservice,id,message\_text). A simplistic block diagram of the system is shown in Figure 1.

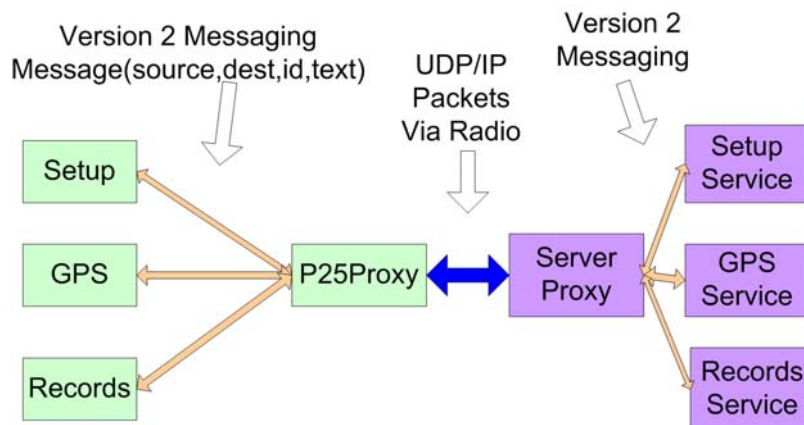


Figure 1. Overview of communication paths

## **P25Proxy message organization**

Each time the P25Proxy application receives a message destined for a server application, P25Proxy adds the message to a queue. The main benefit of the P25Proxy is that it interprets RCP feedback from the radio to determine whether the next message in the queue should be sent, discarded or held off. There are five RCP packet types currently supported. These are: radio power has been turned on (POWER\_UP), data service not registered (NOT\_REGISTERED), packet loss occurred (PACKET\_LOSS), data service not available (SVC\_NOT\_AVAIL) and data service available (SVC\_AVAIL). P25proxy has a designated port to receive RCP packets. To interpret the RCP packets it has received, it simply uses a flagging system. For instance, if a SVC\_NOT\_AVAIL packet is received, the *DataServiceAvailable* flag is set to false. When P25Proxy is about to transmit a message over the radio, it first checks the status of all relevant flags. When the *DataServiceAvailable* flag is false, the message is discarded. The radio sends an RCP packet automatically only when the state changes, thus the proxy application uses a “no news is good news” approach to using RCP to monitor radio status. That is, at startup, P25Proxy assumes that the connection is working properly.

Because there are numerous instances in which the packet should not be sent, this capability is undoubtedly an asset. Data cannot be sent while voice is being transmitted (the radio is in the XMIT state) or received (the radio is in the BUSY state), thus, during this time the next message in the queue is not sent but bumped into a wait cycle. Once in the wait cycle, the *radio\_busy* and *radio\_xmit* flags are checked every second. P25Proxy determines the status of these flags through feedback from the radio application. The packet is sent if there is no traffic in either direction. After a maximum amount of time,

determined by the BusyWait parameter in the registry, if the radio has not been able to transmit the message due to continuous traffic, the message is discarded and the source application is informed.

Other radio states cause the message to be discarded immediately and an error message sent to the source application. These include SVC\_NOT\_AVAIL and NOT\_REGISTERED. The corresponding error messages inform the source application that the radio is not on a data capable channel and that the data registration failed, respectively. Knowing the radio is powered off will also cause the P25Proxy to discard a message and send an error message. However, in this case, the knowledge of the ON/OFF state is not found through the content of an RCP packet or that of radio feedback. While running, the P25Proxy periodically sends RCP requests to the radio control at a rate determined by the parameter KeepAlive in the registry. If a certain amount of time elapses without receiving a response, the radio is presumed to be powered off. The parameter KeepAlive in the registry specifies how large the time elapse must be before power is assumed to be off.

When scan is turned on, sending a data over the radio will automatically cause scan to turn off, but the radio alone will not turn it back on once transmission is complete. To compensate, the P25Proxy has been developed to check the status of scan before a message is sent, and if it is enabled it will turn scan on a certain amount of time after transmission. The amount of time to wait is determined by the ScanHoldoff parameter in the registry. The ScanHoldoff time delay is incorporated so that the radio will be able to receive replies from the server. Data cannot be received by the radio while scan is on and it is possible that a response from the server is expected.

## **Error message generation**

Initially, Project54 messages were designed to be sent between applications on a single computer. Because of the development of proxy applications, the messages may now be sent out over less reliable network links to reach remote services. This increases the likelihood of errors and the need to deal with them. To notify an application that an error has occurred while attempting to send a Project54 message, the P25Proxy application sends a Project54 style message, with the ID: "P54\_MESSAGE\_ERROR". Furthermore, the text of the error message will start with an identifier that elaborates on the type of error that occurred. These include: UNKNOWN\_APPLICATION, MESSAGE\_REJECTED, CONNECTION\_NOT\_READY, and CONNECTION\_FAILURE. The only identifier currently utilized is CONNECTION\_NOT\_READY which applies when the volume of traffic on the radio is too heavy to send the message, data service registration failed or when the radio is not on a data capable channel. At the time of this writing, applications have not been written to recognize or utilize error messages they may receive from the P25Proxy.

## **Server side proxy**

The ServerProxy, the proxy application located at the server, must convert the UDP/IP packets received into the correct Project54 Message format before it can pass them along. The only difference between how this is done in the ServerProxy and a proxy in a car, is the source field. Because multiple cars with the same application names will be communicating to the server, something extra needs to be added to the remote application name so that each car has a unique application name. The ServerProxy

accomplishes this by adding on the IP address and port number from the remote application. Thus, the server applications see dclient:ip.ip.ip:port as the source application as opposed to just dclient. For example, “setup” may become “setup:10.1.9.54:1234”.

## References

- [1] Miller, W. T., “APCO Project 25 RCP in Project54”. ECE.P54.2005.1. March 21, 2005, University of New Hampshire, Consolidated Advanced Technologies for Law Enforcement Laboratory (CATLAB) - Project54.
- [2] Pelhe, A., Kun, A., Miller, W. T., “Project54 system software architecture”. Winter International Symposium on Information and Communication Technologies, Cancun, Mexico, January, 2004.
- [3] Miller, W. T., “Remote Project54 application messaging via the Proxy Application”. ECE.P54.2003.4. March 25, 2003. University of New Hampshire, Consolidated Advanced Technologies for Law Enforcement Laboratory (CATLAB) - Project54.
- [4] Miller, W. T., “Project54 Application Manager messaging”. ECE.P54.2003.2. March 23, 2003. University of New Hampshire, Consolidated Advanced Technologies for Law Enforcement Laboratory (CATLAB) - Project54.