Tests of Xenocode's byte blaster client software with an nbsp-based noaaport system*

José F. Nieves
Department of Physics
University of Puerto Rico
nieves@ltp.upr.clu.edu

Ray Weber
Massachusetts Skywarn
rayoffice@ndws.com

18 Dec 2005

Abstract

This short note summarizes the tests and workarounds that we have done with the Xenocode byte blaster code that would allow it to ingest data from a noaaport-based emwin server.

Introduction and Motivation

nbsp is a software suite to receive and process data from a noaaport system. Among its features, it can redistribute the data masquerading as an emwin (byte blaster) server. Our original goal was to determine to what degree the current version of the common byte blaster client programs for windows could get the data from such a noaaport processing system, with the noaaport side being handled by nbsp. Our tests, using *perlemwin* as the client software (running in linux and freebsd) showed that it should be possible to do so.

It was immediately obvious that there were some issues ingesting the feed with Xenocode's byteblaster emwin client software. After some time, we found several inefficiencies in the byte blaster code that caused it to miss data sent from nbsp. As already mentioned, *perlemwin* under linux or FreeBSD has no issues at all.

We have made fixes to this code to allow it to work at the potential higher data rates that nbsp allows. Testing has showed it to be working rather well. However any emwin software based on this code needs to implement the fixes so it will keep up with higher data rates than the original 9.6 and 19.2 kbit rates.

Our comments below are based on our experience, and our motivation in this short note is to document the changes we had to make and our understanding of the issues involved.

*Authors' names appear in alphabetical order.

Background

The byte blaster code in the client/ENGINE.FRM is based on an event loop. As a result, the data processing portion of the code is called only in response to an event, and the way this event is triggered in the code is by setting a timer (controlled by the variable Timer1 in the code). When the timer expires, the call-back function is called and the program goes to process a block of data (1116 bytes). After processing one block, the timer is reset and the program waits until the timer expires again to enter the next processing loop, and so on.

Problem

The problem is that time interval defined in the code for the Timer1 is 500 ms (milliseconds). In particular, this means that if it is reading from a noaaport based server, it will miss most of the data sent to it, since the data is sent at a much faster rate than a few kbytes/second. Usually, products comprised of just one block would be received and processed, but products made out of more than one block would be received incomplete since some of the blocks would be missed in the 500 ms waiting interval.

Workaround

The best workaround we found was to set the Timer1 as short as possible, 1 ms. There is little doubt that this is not "the best" solution. But, from an operational point of view, it is the least disruptive modification of the code that allows it process the data as fast as possible. As already mentioned, for practical purposes, it seems to work rather well.

Nevertheless, that change by itself was not enough. Changing that interval value had other effects, related to buffering and timeout waiting for data. Fortunately, it seems that the additional changes required are restricted to modify the size of a receive buffer and the timeout setting for "no data received". A suitable increment of the values of both parameters seems to produce a perfectly working configuration.

We are not aware of any other side effects of those modifications. The recompiled byte blaster client has been tested using nbsp running on linux as the noaaport server, and also under freebsd, independently, and we have not found any further problems.

Summary

We have tested Xenocode byteblaster emwin client software against an nbsp based emwin-like server. We have identified some inefficiencies in the byte blaster code that causes it to miss data sent from nbsp, and we have suggested specific modifications of the code as workarounds.

We are publishing openly the source of the only modified file in the client software "ENGINE.FRM", as well as a diff file "ENGINE.FRM.diff" against the original file, for documentation purposes. We do so with the intention that authors of byte blaster-based windows emwin software can use this code as a reference point to modify and probably optimize their code within their own framework, that would allow them to work with an nbsp-based noaaport system.