BEDs Bench Testing and ZMODEM Tuning

Eric Martin CCE 901023 MBARI

April 7th, 2016

Contents

1 Summary

Since deployment of the BEDs experiments, recovering data stored subsea in post-event sensors has proven difficult. Compounding the normal issues of acoustic communication in seawater, BEDs that have experienced a benthic event are often buried under multiple meters of sediment and at the base of a canyon. This presents issues of added acoustic impedance, as well as propagation geometry that lead to multi-path acoustic interference. It has been observed, however, that even in seemingly good acoustic conditions, there were still problems in transmitting data back to the wave-glider and to the Benthos UDB from the R/V Paragon. After many attempts from the R/V Paragon to download data, it was concluded that some tuning of the file transfer might improve performance.

The purpose of this test was to reproduce the conditions observed offshore and develop strategies that improve robustness of throughput, but also improve error recovery. ZMODEM is the protocol under test. There is an option to use the Kermit Protocol, but due to the possible crash recovery features of ZMODEM, it has been the primary system under test.

ZMODEM should be a compatible protocol for use in acoustic communications. Benthos, who manufactures our modem units, even employs ZMODEM for their datalogger applications. Thus, after observations offshore, we pose that a better understanding of the interoperation of this protocol through the acoustic modem transceiver might lead to successful set of configurations for optimized data transmission.

2 Testing

2.1 Test Setup

The bench test modeled at-sea hardware for use aboard the R/V Paragon. The bench test and programming fixture was an ideal representation of the relevant hardware in the BEDS, replicating that end of the sub-sea system. The topside equipment, a Benthos UDB-9400E and LF "dunker" were placed on the bench next to the BEDs electronics. A laptop was connected serially to both devices' console ports. Monitoring the auxiliary serial connection of the BEDs facilitates the observation of real-time syslog-like messages during file transactions.

2.2 Benthos Modem Variables

- *Acoustic Bit Rate:* `S4` is a register controlling the *transmit* data rate through the acoustic modem. Playing with this value is more impactful than just modifying the speed transmission, as each is mapped to a different strategy for convolutional coding (1/2 for all rates below 10,240 bps). The other impact of changing acoustic bit rate is that the multipath guard period is set by this same value. Multi-path guard is only used at or below 1066bps.
- *Forwarding Delay:* `S8` is a register polled by the BED at the start of a ZMODEM interaction. It is the time between entry of data on the RS232 serial port and start of buffer transmission through the acoustic transducer.
- *Static Configurations:* The rest of the values will be held constant at the settings in Tables 1 and 2 below.

Table 1: Topside modem register settings.

S Reg	Value	Name
S0	086	Version
S1	000	Doppler Off
S2	000	Packet Retry Off
S3	002	Serial BR 9600 no echo
S4	005	ACOMM BR VARIABLE
S5	000	Not Used
S6	001	Transmit Power Low
S7	015	Timeout (not ato relev)
S8	010	Forwarding Delay 0.5s
S9	001	UDB Copressor Installed
S10	255	Idle time not used
S11	000	No RS232 Flow Control
S12	129	ATX test 32-byte w/echo
S13	000	Verbosity off
S14	003	Remote 7 (ATD Changes)
S15	001	Op mode (for ats?)
S16	001	Error packets displayed
S17	001	Lowpower wakeup inserted
S18	000	Modem Address 0
S19	000	??
S20	000	??

Table 2: Subsea modem register settings.

S Reg	Value	Name
S0	042	Version
S1	000	Doppler Off
S2	000	Packet Retry Off
S3	002	Serial BR 9600 no echo
S4	005	ACOMM BR VARIABLE
S5	000	Not Used
S6	006	Transmit Power -6dB
S7	015	Timeout (not ato relev)
S8	010	Forwarding Delay 0.5s
S9	001	UDB Copressor Installed
S10	036	Idle time = 180s
S11	000	No RS232 Flow Control
S12	000	ATX test disabled
S13	000	Verbosity off
S14	000	Remote 0 (ATD Changes)
S15	001	Op mode (for ats?)

S Reg	Value	Name
S16	001	Error packets displayed
S17	001	Lowpower wakeup inserted
S18	007	Modem Address 7
S19	000	??
S20	000	??

2.3 Software Variables

2.3.1 Transmit Variables

Using ZMODEM on picodos, we are extremely limited on our options for file transfer. We can, however modify the size of the sliding window that ZMODEM uses and the length of packets transmitted. Naturally, by modifying the packet length, we expect an increase total file transmission in an ideal scenario (direct IO), but this should also help in fewer errors, and faster recovery times.

To simplify the tests, we shall set the windowsize (`/W`) and the frame length (`/L`) to the same values. The window size must be greater than 256 and the frame length can vary from 32 to 1024 bits.

The transmit command is:

```
zs /V0 /W<x> /L<y>
```

2.3.2 Receive Settings

For this test I am employing minicom, which uses the unix lrzsz library for data transfer. This library is available and likely easy to integrate if not already available on the wave glider. The command flags optioned in are increased verbosity (`-vv`), binary transfer (`-b`) and the ability to resume partial files (`-r`). Additionally, for low baud rate transmission, I recommend increasing the timeout of the RZ as well, by using the `-t <s/10>` flag, with an integer following in tenths of seconds. Other flags are `Z` for ZMODEM protocol, and `U` for unrestricted mode. The rz command is thus:

```
rz -ZbUrvv -t 200
```

I recommend use of a tool like minicom only because is it effective at automatically transitioning from a terminal application to the above call to ZMODEM when a `ZRINIT` message arrives.

2.4 Test Constraints

This test shall evaluate success by measuring time of transmission from execution of the zs on the BED. We are using the same test file, `1.WAT`, which is about 100KB in size. These transfers with establish a baseline measures on transmission times through the 9600 kbps serial line, varying the other 4 variables in succession. User stimulated restarts will be annotated if required.

2.5 Test Procedure

For each test, after successfully achieving acoustic communications between top- and bottom-side gear, a file transfer of `1.WAT` will be initiated through the varied `ZS` call. Initial tests will be through the auxiliary serial port, and then similar tests will be conducted through the acoustic link. Iterative changes of the 4 parameters will be varied until stable transfers are made.

3 Results

3.1 Direct to Motherboard

In Table 3 tests were direct PC to BEDs motherboard RS232 connections, just to serve as a baseline for non-acoustic transfer rates.

Table 3: Direct RS232 9600 BPS Connection Download Tests

ZS CMD	Window	Packet Sz	Time (s)	Bytes Xmitted	kbps
zs /W256 /L32	256	32	27.5	100KB	29
zs /W256 /L256	256	256	17.2	100KB	47
zs /W256 /L512	256	512	14.9	100KB	54
zs /W256 /L1024	256	1024	14.3	100KB	56
zs /W512 /L32	512	32	27.1	100KB	30
zs /W512 /L256	512	256	16.2	100KB	49
zs /W512 /L512	512	512	15.1	100KB	53
zs /W512 /L1024	512	1024	14.3	100KB	56
zs /W1024 /L32	1024	32	27.1	100KB	30
zs /W1024 /L256	1024	256	16.9	100KB	47
zs /W1024 /L512	1024	512	15.0	100KB	53
zs /W1024 /L1024	1024	1024	14.4	100KB	56

3.2 Using Benthos Acoustic Modems

As previously described, the modem tests best replicate the at-sea hardware configuration when working from the R/V Paragon. By being able to see the syslog traffic on the secondary RS232 port of the BED motherboard, diagnosis of problems was much more straightforward. I started by varying the search parameters to better understand their impact on transfers. Initial tests were done without adding interruptions in communication. Simply by characterizing maximum throughput, I was able to determine maximum and minimum conditions for the 4 test variables to ensure reliable transmission. We know that these variables are not mutually exclusive, and by varying them thoroughly, I was able to create conditions that previously observed modes of error seen offshore were yet unattributed to.

3.3 Modes Of Failure and Their Causes

- *ZModem Timeouts*: These are messages seen when the receiving application does not receive packets in a determined amount of time. The defaults for these in both Hyperterminal and the LRZSZ package are set for times commensurate with transfer speed up at 9600 bps, and we are operating closer down to 600 bps. Zmodem should recover from a timeout successfully, but in order to accomplish this, the receiving software sends a new request for more data, often times this request could occur while data is being received, and this is a violation of half-duplex communication enforced by the acoustic modem. Another cause of a timeout would also be a forwarding delay that was configured to be so large that the timeout would trip.
- *Loss of Communication with BED*: After a failed ZMODEM download, there is a period of time when the BED is still running ZS.pxe and will only respond to well formed ZMODEM messages. However, topside you might be attempting to illicit a `BEDS>` prompt, and would get no response.
- *Bad CRC*: This is a message seen in verbose instances of `rz`, these messages occur when packets arrive on the serial port and do not pass inspection. The cause of this has been attributed to ZMODEM frames arriving in two acoustic packets.
- *Poor Acoustic Communications*: The environment can cause packet to be mishandled. Since the default mode of error correction on the modem is to not pass data that does not meet checks, it can appear to DTE and DCE devices like no communication is happening at all. This lack of communication will manifest as timeouts as well.

3.4 Resultant Baud Rates With Successful Transfers

The characterization of the failures led to a better understanding of the interdependency of the configuration parameters available to this interchange. Though many configurations were attempted (see Appendix A), below find a small table that summarizes successful transfer schema. For all of these settings, it should be noted that the forwarding delay was set to 0.5 seconds. I believe that transfers could perform at lower values of this setting, but then manual entry commands would no longer function. This setting works for both human driven keyboard entry without hampering an automated interaction, such as ZMODEM.

Also, scenarios listed below combine settings in such a way that minimizes the impact of transmitting through acoustic modems. When packets are set larger than specified at listed baud rates in Table 4, resultant ZMODEM packets are split into two acoustic messages. The receiving ZMODEM software does often will time out and evaluate a packet that is half received, and as such will return `Bad CRC` error messages.

Table 4: Successful combinations of acoustic baud rates and packet lengths sent by ZMODEM.

Acoustic Baud Rate	`zs` Length	Xmit Duration	Effective Transfer Speed
1200 bps	1024 B	12 s	~65 bps
1066 bps	1024 B	10 s	~54 bps
800 bps	1024 B	8 s	~45 bps
600 bps	512 B	~8s	~32 bps
300 bps	256 B	~8s	~14 bps
140 bps	128 B	~8s	~7 bps

4 Recommendations

From the above results, we can recommend certain configurations that will provide consistent success in amenable acoustic environments. The solution is difficult to capture or automate, as you must be sure you have all four separately configured devices tuned properly. This can best be described, perhaps, in a procedure to follow. Once you have the modems and transport software configured, you can make adjustments based on environment impact to achieve success.

4.1 Procedure

4.1.1 Configure the Modems

1. You can simply reflect the settings presented in Tables 1 and 2 to begin with.

2. Start with a low baudrate, 600 bps.

3. Adjust both the top and bottom power settings, `S6`, until you can validate successful communciations. This is either done by using the link test `ATY`n in command mode or by simply illiciting the `BEDS>>` prompt in online mode.

   - **WARNING**: It is possible to set your power too high, acoustic messages can clip and this has been observed in the field.

4. Once communications are set, verify a quick forwarding delay (<0.5s). Also verify that modem verbosity is off, and that you disable the error check. We want the modem to pass even garbled messages for a few reasons:

   - By receiving even a bad packet, the ZMODEM receiver will see that packet and immediately request a retransmission of the packet.
   - ZMODEM is sophisticated enough in some cases to be able to tell if some of the packet is good and save only that part.
   - It keeps the interaction alive and determined, versus both sides waiting for timeouts.

5. Be sure to exit command mode and lengthen the timeout of the BEDs to avoid missing your window while you configure the rest of your download.

4.1.2 Form your command to the BEDS for Download

You can use whatever terminal application you like, but I reccommend minicom on POSIX systems, as it was used successfully for all these tests. Also, Minicom has a scripting interface and automatic ZMODEM transfer for quickest receive response.

1. Use whatever means you choose to determine which file you are downloading. Except avoid using the `event` script because that will initiate the ZMODEM connection with default packet length parameters that will result in potential problems at lower acoustic baud rates.

2. Once you have decimated or selected your files, create a script or a clipboarded command somewhere that you can execute or paste into your terminal window to the topside modem. As in Section 2.3.1, this should be a `ZS` call with both the `/W` and `/L` variables set to match values at or less than what is listed for your selected acoustic baud rate in Table 4.

3. Configure the ZMODEM receive call, in whichever terminal program you are using. This should not need to change, and you can follow the prototype shown in Section 2.3.2.

4.1.3 Begin the Download

1. Paste or execute the command for `ZS` into the terminal in online mode, and then wait for the subsea electronics to initiate the transfer.
2. Keep in mind that the subsea electronics polls two registers of a modem at address 0 before it starts the transmission. If this fails, the ZMODEM transfer may not start for up to a minute.
3. Once you see the ZRINIT header, which starts with a `**B0`, you can initiate the transfer, or let minicom pick it up automatically.

4.1.4 Troubleshooting Transfers

- Occasionally, bad packets may arrive, in this case, the receiver will issue a new request for the same packet and the protocol should recover.
- Often, one side will miss a messsage altogether. In this case, you should wait for the receiver to timeout and request a new packet. This timeout is set in you receive command flags.
- If communications is lost for a significant amount of time, a value hard to evaluate, the ZMODEM software on the BEDS can give up and return from the transmission with failure. However, this return is not to the command prompt normally seen. ZMODEM is still running, and as such the topside cannot illicit the `BEDS>>` prompt. If this seems the case, try to manually initiate a new ZMODEM receive command topside, and if the BEDS gets that command, it will return a `ZACK` with a notification that the transfer is over and incomplete. You should then get a prompt back.
- Often both sides can lose sync and they will both timeout. If this happens just restart the transfer, and if any data has been written to your file top-side, crash recovery should resume the transfer.
- If incoming data seems clouded by errors and bad CRC's, I advise going all the way back to re-evaluating the modem connection, and changing acoustic baud rates and power as necessary. Remember to adjust packet length if you change baud rate.

4.2 Outstanding Issues

4.2.1 Future BEDS Releases

1. Modify the events program to allow `ZS` packet length settings to pass if they don't already.
2. Consider removing the echo functionality on the acoustic serial interface, this can cause issues when the forwarding delays are set to be low values.

4.2.2 Top-side Issues

1. *ZMODEM Resume On the Waveglider*: At present the ability to resume crashed file downloaded when using the waveglider is hindered. The lower limit of writing buffered data out to file on the lrz command built there is around 4KB, and should be much shorter. This is a protection set in normal conditions to reduce file i/o, but many files available from the BEDs are only 10-20KB in the first place. A lower value would be preferable.