NORM Pre-Coder (*npc*) Usage Notes

Background

The NACK-Oriented Reliable Multicast (NORM) protocol is capable of supporting robust transmission of content to "silent" receivers that are required or only capable of operating in an emission-controlled (EMCON) manner. This capability is enabled when the NORM sender is configured to proactively transmit Forward Error Correction (FEC) erasure coding content as part of its original data transmission. For NACK-based operation, the FEC repair packets are usually sent only reactively, in response to repair requests (NACKs) from the receiver group. However, hybrid operation with a combination of proactive FEC content and additional reactive FEC repairs as needed is also supported. Similarly, a mix of nacking and silent receivers may be supported with silent receivers capitalizing on the FEC repair information sent proactively and/or reactively. The purpose of the NORM Pre-Coder (npc) software utility described here is to support additional robustness for purely-proactive sessions, where the receivers are unable to request repair or retransmission of content.

The Naval Research Laboratory (NRL) reference implementation of the NORM protocol includes support for 8-bit (and very soon 16-bit) Reed-Solomon FEC encoding with additional support for other coding algorithms (e.g., Low-Density Parity Check (LDPC)) planned for the future. The NORM specification allows for different FEC algorithms to be applied within the protocol. The current Reed-Solomon NORM FEC algorithms in the NRL implementation are limited to modest code block sizes (With 16-bit Reed-Solomon coding, larger block sizes will be allowed but very high data rates may not be possible). For channels with random errors, the current NORM FEC codecs are often adequate as there is flexibility in how the encoded data can be partitioned into FEC blocks (a block consists of some number of data segments (packets)) and the number of FEC parity packets that can computed and possibly transmitted per source data block. For channels with large bursts of packet loss (with respect to the configured NORM FEC block size), it is quite possible that the number of lost packets (erasures) that occur within a NORM FEC block may exceed the configured erasure-filling capability. The *npc* utility was created to "pre-encode" (and "post-decode") files for NORM transmission to silent (non-NACKing) receivers by adding additional FEC encoding, and importantly, interleaving of the FEC segments (packets) to re-distribute bursts of packet losses as random losses over the entire file. It is thus most applicable to very large files (with respect to FEC block sizes).

Overview

The *npc* utility takes, as input", a file and logically divides it into segments, adding cyclic-redundancy checksum (CRC) to the segments, encoding the source segments with Reeed-Solomon encoding (adding a configurable number of parity segments per FEC source block), and interleaves the source and encoding segments to an output file. The use of the CRC allows erasure to be detected and also provides additional assurance of correct content delivery by possibly detecting bit errors that may have been undetected during transport (i.e., link-layer framing, Internet Protocol (IP), and/or User Datagram

Protocol (UDP) checksums). The interleaving by default is a block interleaver using the entire file as a logical block, but a limit on the interleaving size can be set to help increase the speed of the *npc* encoding/decoding process. This may be useful for extremely large file sizes.

Usage and Examples

The following is a synopsis of *npc* usage:

```
npc {encode|decode} input <inFile> [output <outFile>]
    [segment <segmentSize>][block numData][parity numParity]
    [imax <widthLimit>][ibuffer <bytes>]
    [background][help][debug <debugLevel>
```

The *npc* utility may be instructed to either "`encode`" a file (add FEC content and interleaving to the given <`inFile`>) or "`decode`" a file that was previously encoded with *npc*. The "`.npc`" file extension is suggested to delineate files that are of the *npc* encoded format. Note the "`output`" filename is optional. By default, npc will use the filename of the <`inFile`> as the output filename, replacing the '`.`' extension delimiter with a '`_`' (underscore) and adding the "`.npc`" extension suffix. The npc format includes some minimal "meta-data" in the first encoded <segmentSize> to convey the file size and name of the original file. On decoding, if the "`output`" file option is omitted, this "meta-data" is used to name the decoded output file.

The optional FEC parameters, <`segmentSize`>, <`numData`>, and <`numParity`> control the logical segmentation, blocking, and amount of FEC parity content added to the file. For use with NORM, it is recommended that the <`segmentSize`> value correspond to the same segmentation size used for NORM transmission. The <`numData`> (source segments per FEC encoding block) and <`numParity`> parameters should be selected to provide erasure filling coverage for the expected transmission packet loss characteristics. Note that when used with proactive NORM FEC transmission, the *npc* encoding provides an "inner" FEC code and interleaving and the NORM protocol provides an "outer" FEC encoding. The "outer" NORM code might be configured to deal with typical random packet loss due to channel BER, etc and the "inner" *npc* interleaving and coding could be correspondingly configured to handle expected burst losses (e.g. outages) that might occur.

To encode a file name "`originalFile.txt`" with the default *npc* naming convention, FEC, and interleaving parameters, use the following syntax:

```
npc encode input originalFile.txt
```

This will produce and output file named "`originalFile_txt.npc`" in the current working directory,

The original file can be recovered (decoded) using the syntax:

```
npc decode input originalFile_txt.npc
```

This will decode the "`.npc`" file, and in this case produce a file named "`originalFile.txt`" in the current working directory. (The file name information

was stored in first segment of the "`.npc`" file). This default naming convention can be overridden by using the *npc* "`output`" option. For example, the syntax:

```
npc decode input originalFile_txt.npc output file.txt
```

This will produce a file named "`file.txt`" that is identical in content to "`originalFile.txt`".

Notes

The FEC and interleaving parameters that are used for *npc* encoding MUST be exactly matched to successfully decode the encoded file. I.e., if the defaults are used for encoding, the defaults must be used for decoding. The parameters that must match include <`segmentSize`>, <`numData`>, <`numParity`>, and <`widthMax`>.

It is possible that in some cases it may be beneficial to apply more proactive FEC content with the *npc* program instead of with the NORM transport. The trade-offs are scenario-specific.

The NRL "*norm*" demonstration application now has options included to support transport of *npc* encoded files. The distinction here is that a file that _fails_ NORM transport might still be successfully decoded with *npc*. There are two receiver-side *norm* demo application options that apply here:

1) The "`saveAborts`" option causes norm to not delete (and attempt to post-process) "aborted" files (files that failed reliable NORM transport).

2) The norm "`lowDelay`" option should be applied for silent-receivers to more immediately deliver "failed" files to the application for post-processing (i.e., attempted *npc* decoding)

npc Command Reference

The following table describes each of the *npc* command-line options

`encode | decode`	Determine whether *npc* is to encode or decode the given <`inFile`>. This option is required and only one should be given.
`input <inFile>`	Specifies the file to be processed. Required option.
`output <outFile>`	Specifies the name of the output file to be produced. Overrides the default *npc* naming convention. Optional.
`segment <segmentSize>`	Sets the segmentation size (packet size) in bytes. Four bytes of the <`segmentSize`> are used for a 32-bit CRC that *npc* applies to each segment. (Default is 1024 bytes)
`block <numData>`	Specify the number of source data segments (packets) per *npc* FEC coding block. (Default is 98 segments).
`parity <numParity>`	Specify the number of FEC parity segments (packets) added per *npc* FEC coding block. (Default is 2 segments).
`imax <widthMax>`	Limits interleaving of encoded file to a maximum interleaver width of <`widthMax`> segments. A value of ZERO (or less) defaults to *npc* calculating a block interleaver that encompasses the entire encoded file size. For extremely large files, this option may be beneficial to limit file seeking operations required to interleave the file. If the encoded file size is less than `<widthMax>*<widthMax>` segments, *npc* will again calculate its own maximum block size. (Default is 1000 segments interleaver depth (i.e., about 1Gbyte interleaver size with the 1024 byte <`segmentSize`> value))
`ibuffer <bufferSize>`	This sets the maximum memory (in bytes) that *npc* allocates for encoding. A larger value allows *npc* to perform file input/output with less seeking and improved encoding/decoding times can be achieved. (Default is 1.5 GByte)
`background`	Runs *npc* as a background process with no console window (Win32).
`debug <debugLevel>`	Specifies debug output verbosity. Higher number is more verbose debugging information. (Default is ZERO).
`help`	Displays *npc* usage.