NAME

LDPCFecScheme −

SYNOPSIS

```
#include <ldpc_scheme.h>
```

Inherits **LDPCFecSession**.

Public Member Functions

LDPCFecScheme ()
˜LDPCFecScheme ()
ldpc_error_status DetermineSymbolSize (INT64 objectSize, int pktSize, int *symbolSize, int *nbSourceSymbols)
ldpc_error_status InitScheme (int symbolSize, int pktSize)
ldpc_error_status BuildPkt (int pktIdx, void **pktBuffer, void *symbol_canvas[], int *ESIofFirstSymbol)
ldpc_error_status DecomposePkt (void *pktBuffer, int ESIofFirstSymbol, void **GeneratedSymbols[], int *ESIofSymbols[])
ldpc_error_status DecodingStepWithPkt (void *symbol_canvas[], void *pktBuffer, int ESIofFirstSymbol, bool store_symbol)
ldpc_error_status DecodingStepWithPkt (void *symbol_canvas[], void *pktBuffer, int ESIofFirstSymbol)
int **getNbSymbolsPerPkt** (void)
int **getNbSourcePkts** (void)
int **getNbParityPkts** (void)
int **getNbSourceSymbols** (void)
int **getNbParitySymbols** (void)

Detailed Description

Class that implements parts of the LDPC-Staircase/Triangle FEC Scheme, as defined in draft-ietf-rmt-bb-fec-ldpc-01.txt (or later version). It defines the notion of packet, i.e. the grouping of several symbols in the same transmission unit. Depending on the initialization, the LDPCFECScheme class can either define internally the optimal number of symbols per packet, or take it as a parameter. Using this class makes the symbol(s) <=> packet mapping almost transparent to the user. In that case, packet creation (SENDER) and packet processing (RECEIVER) are completely managed by this class (e.g. there is no need to call the LDPCFECSession::DecodingStepWithSymbol() method any more).

When **LDPCFecSession** and LDPCFecScheme are both used, the **LDPCFecSession** MUST be initialized first (with **InitSession()**), THEN the LDPCFecScheme (with **InitScheme()**).

Constructor & Destructor Documentation

LDPCFecScheme::LDPCFecScheme ()

LDPCFecScheme Constructor and Destructor.

LDPCFecScheme::˜LDPCFecScheme ()

Member Function Documentation

ldpc_error_status LDPCFecScheme::DetermineSymbolSize (INT64 objectSize, int pktSize, int * symbolSize, int * nbSourceSymbols)

Determine the optimal symbol size when the object size and packet size are both known.

This function defines the optimal symbol size in order to maximize the erasure recovery efficiency. The actual number of symbols per packet can then be retrieved by means of the **getNbSymbolsPerPkt()** function. This value must be an integer, i.e. that the packet size must be a multiple of the resulting symbol size (which also depends on the object size). It usually means that the packet size must be a multiple of a certain power of 2. If not possible, an error is returned.

When the object size is not multiple of the symbol size, it is left to the application to handle the possible padding of the last source symbol.

Parameters:

objectSize (IN) the object size (bytes).
pktSize (IN) the packet size (bytes). Depending on the number of symbols per packet, the packet size must sometimes be a multiple of 4 or 8.
symbolSize (OUT) opimal symbol size determined by this function.
nbSourceSymbols (OUT) corresponding number of source symbols.

Returns:
Completion status (LDPC_OK or LDPC_ERROR).

ldpc_error_status LDPCFecScheme::InitScheme (int symbolSize, int pktSize)

Initialize the LDPC scheme. Note that the packet size must be a multiple of the symbol size.

Parameters:
symbolSize (IN) the symbol size, probably calculated by the **DetermineSymbolSize()** function above.
pktSize (IN) the packet size (bytes).

Returns:
Completion status (LDPC_OK or LDPC_ERROR).

ldpc_error_status LDPCFecScheme::BuildPkt (int pktIdx, void ** pktBuffer, void * symbol_canvas[], int * ESIofFirstSymbol)

Build the packet of the index from an appropriate number of symbols. Used by a sender. There are always **getNbSymbolsPerPkt()** symbols per packet, even when the number of source or repair symbols are not multiple of the number of symbols per packet. Note that only homogeneous packets are created, i.e. packets consisting either of source symbols (AKA source packets), or or of repair symbols (AKA repair packets).

Parameters:
pktIdx (IN) Index of the packet to build, in [0; getNbPkts() - 1] range.
pktBuffer Data buffer where the packet should be written.
ESIofFirstSymbol (OUT) ESI of the first symbol chosen to be included in this packet.

Returns:
Completion status (LDPC_OK or LDPC_ERROR).

ldpc_error_status LDPCFecScheme::DecomposePkt (void * pktBuffer, int ESIofFirstSymbol, void ** GeneratedSymbols[], int * ESIofSymbols[])

Split a received packet into the set of its constituting symbols.

Parameters:
pktBuffer (IN) Data buffer containing the packet received.
ESIofFirstSymbol (IN) ESI of the first symbol of the packet.
GeneratedSymbols[] (OUT) table containing pointers to all the symbols of the packet, including the first one. There are always **getNbSymbolsPerPkt()** entries. This table is allocated by the function and must be free'ed by the caller.
ESIofSymbols (OUT) table containing the ESI of all the symbols of the packet, including the first one. There are always **getNbSymbolsPerPkt()** entries. This table is allocated by the function and must be free'ed by the caller.

Returns:
Completion status (LDPC_OK or LDPC_ERROR).

ldpc_error_status LDPCFecScheme::DecodingStepWithPkt (void * symbol_canvas[], void * pktBuffer, int ESIofFirstSymbol, bool store_symbol)

Perform a new decoding step thanks to the newly received packet. This is the same as **LDPCFecSession::DecodingStepWithSymbol()** but with a packet as input rather than a symbol.

Parameters:

symbol_canvas (IN-OUT) Global array of received or rebuilt source symbols (parity symbols need not be stored here). This is a table of k pointers to buffers. This array must be cleared (memset(0)) upon the first call to this function. It will be automatically updated, with pointers to symbols received or decoded, by this function.
pktBuffer (IN) Pointer to the buffer containing the new packet.
ESIofFirstSymbol (IN) ESI of the first symbol of the packet.
store_symbol (IN) true if the function needs to allocate memory, copy the symbol content in it, and call any required callback. This is typically done when this function is called recursively, for newly decoded symbols, or under special circunstances (e.g. perftool).

Returns:

Completion status (LDPC_OK or LDPC_ERROR).

ldpc_error_status LDPCFecScheme::DecodingStepWithPkt (void * symbol_canvas[], void * pktBuffer, int ESIofFirstSymbol)

Perform a new decoding step thanks to the newly received packet. Same as the other DecodingStepWithSymbol method, without the store_symbol argument (prefered solution).

Parameters:

symbol_canvas (IN-OUT) Global array of received or rebuilt source symbols (parity symbols need not be stored here). This is a table of k pointers to buffers. This array must be cleared (memset(0)) upon the first call to this function. It will be automatically updated, with pointers to symbols received or decoded, by this function.
pktBuffer (IN) Pointer to the buffer containing the new packet.
ESIofFirstSymbol (IN) ESI of the first symbol of the packet.

Returns:

Completion status (LDPC_OK or LDPC_ERROR).

int LDPCFecScheme::getNbSymbolsPerPkt (void) `[inline]`

Return the number of symbols that are grouped in the same packet (AKA symbol group). There are always this number of symbols per packet, even when the number of source or repair symbols is not multiple of the number of symbols per packet.

Returns:

Number of symbols grouped in any packet.

int LDPCFecScheme::getNbSourcePkts (void) `[inline]`

Return the number of source packets (i.e. consisting only of source symbols). It might differ from the number of source symbols since several symbols can be grouped in the same packet.

Returns:

Number of source packets.

int LDPCFecScheme::getNbParityPkts (void) `[inline]`

Return the number of parity packets (i.e. consisting only of parity symbols). It might differ from the number of parity symbols since several symbols can be grouped in the same packet.

Returns:

Number of parity packets.

int LDPCFecScheme::getNbSourceSymbols (void) `[inline]`

Return the number of source symbols It might differ from the number of parity packets since several symbols can be grouped in the same packet.

Returns:

Number of parity packets.

int LDPCFecScheme::getNbParitySymbols (void) `[inline]`

Return the number of parity symbols It might differ from the number of parity packets since several symbols can be grouped in the same packet.

Returns:

Number of parity packets.

Author

Generated automatically by Doxygen for ldpc from the source code.