Model Based Approach to Evaluate the Efficiency of FEC Coding Algorithm

Authors
Ashwani Tayal¹, Kamal Saluja²
School of PG Engineering, Bahadurgarh
Email-Love2ashwani@gmail.com

ABSTRACT
Many wireless communication systems such as IS-54, enhanced data rates for the GSM evolution (EDGE), worldwide interoperability for microwave access (WiMAX) and long term evolution (LTE) have adopted low-density parity-check (LDPC), tail-biting convolutional, and turbo codes as the forward error correcting codes (FEC) scheme for data and overhead channels. Therefore, many efficient algorithms have been proposed for decoding these codes. However, the different decoding approaches for these two families of codes usually lead to different hardware architectures. Since these codes work side by side in these new wireless systems, it is a good idea to introduce a universal decoder to handle these two families of codes. The present work exploits the parity-check matrix (H) representation of tailbiting convolutional and turbo codes, thus enabling decoding via a unified belief propagation (BP) algorithm. Indeed, the BP algorithm provides a highly effective general methodology for devising low-complexity iterative decoding algorithms for all convolutional code classes as well as turbo codes. While a small performance loss is observed when decoding turbo codes with BP instead of MAP, this is offset by the lower complexity of the BP algorithm and the inherent advantage of a unified decoding architecture.

EXISTING SYSTEM
For analysis purposes the packet-loss process resulting from the single-multiplexer model was assumed to be independent and, consequently, the simulation results provided show that this simplified analysis considerably overestimates the performance of FEC.

Evaluation of FEC performance in multiple sessions was more complex in existing applications.

Surprisingly, all numerical results given indicates that the resulting residual packet-loss rates with coding are always greater than without coding, i.e., FEC is ineffective in this application.

The increase in the redundant packets added to the data will increase the performance, but it will also make the data large and it will also lead to increase in data loss.

PROPOSED SYSTEM
In this work we have evaluated the performance of FEC coding more accurately than previous works. We have reduced the complexity in multiple sessions and introduced a simple way for its implementation.

We show that the unified approach provides an integrated framework for exploring the tradeoffs between the key coding parameters: specifically, Interleaving depths, channel coding rates and block lengths.

Thus by choosing the coding parameter appropriately we have achieved high performance of FEC, reduced the time delay for Encoding and Decoding with Interleaving.
MODULE DESCRIPTION

1. FEC Encoder
FEC is a system of error control for data transmission, where the sender adds redundant data to its messages. This allows the receiver to detect and correct errors (within some bounds) without the need to ask the sender for additional data. In this module we add redundant data to the given input data, known as FEC Encoding. The text available in the input text file is converted into binary. The binary conversion is done for each and every character in the input file. Then we add the redundant data for each bit of the binary. After adding we have a block of packets for each character.

The User Interface design is also done in this module. We use the Swing package available in Java to design the User Interface. Swing is a widget toolkit for Java. It is part of Sun Microsystems’ Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs.

2. Interleaver
Interleaving is a way of arranging data in a non-contiguous way in order to increase performance. It is used in data transmission to protest against burst errors. In this module we arrange the data (shuffling) to avoid burst errors which is useful to increase the performance of FEC Encoding. This module gets the input as blocks of bits from the FEC Encoder. In this module we shuffle the bits inside a single block in order to convert burst errors into random errors. This shuffling process is done for each and every block comes from the FEC Encoder. Then we create a Socket connection to transfer the blocks from Source to the Queue. This connection is created by using the Server Socket and Socket class Available in Java.

3. Implementation of the Queue
In this module we receive the data from the Source system. This data is the blocks after FEC Encoding and Interleaving processes are done. These blocks come from the Source system through Server Socket and Socket. Server socket and Socket are classes available inside Java. These two classes are used to create a connection between two systems inside a network for data transmission. After we receive the packets from Source, we create packet loss. Packet loss is a process of deleting the packets randomly. After creating loss we send the remaining blocks to the Destination through the socket connection.

4. De-Interleaver
This module receives the blocks of data from the Queue through the socket connection. These blocks are the remaining packets after the loss in the Queue. In this module we re-arrange the data packets inside a block in the order in which it is before Interleaving. This process of Interleaving and De-Interleaving is done to convert burst errors into random errors. After De-Interleaving the blocks are arranged in the original order. Then the data blocks are sent to the FEC Decoder.

5. FEC Decoder
This module gets the input from the De-Interleaver. The received packets are processed to remove the original bits from it. Thus we recover the original bits of a character in this module. After retrieving the original bits, we convert it to characters and write it inside a text file.

6. Performance Evaluation
In this module we calculate the overall performance of FEC Coding in recovering the packet losses. After retrieving the original bits, we convert it to characters and write it inside a text file. This performance is calculated by using the coding parameters like Coding rate, Interleaving depth, Block length and several other parameters. First we calculate the amount of packet loss and with it we use various formulas to calculate the overall performance of Forward Error Correction in recovering the network packet losses.
SCOPE OF THE PROJECT
In this project we are going to evaluate the efficacy of FEC coding. To evaluate it we are going to transfer data from Source to Destination. Before receiving the data in the Destination we create packet loss in Queue. Thus after creating packet loss, we receive the remaining packets in the Destination. Then we recover the lost Packets in the Destination and evaluate FEC’s performance.

MODULE DIAGRAM
FEC Encoder

Input File → Binary Conversion → Packet Separation → Adding Redundant Bits → Packets after FEC Coding

Interleaver

Packets after FEC Coding → Shuffling the packets inside a block → Packets after Shuffling

Implementation of the Queue

Packets from Interleaver → Packet loss → Packets after loss
De-Interleaver

CLASS DIAGRAM

Class: A Class is a description for a set of objects that shares the same attributes, and has similar operations, relationships, behaviors and semantics. Generalization: Generalization is a relationship between a general element and a more specific kind of that element. It means that the more specific element can be used whenever the general element appears. This relation is also known as specialization or inheritance link.

Realization: Realization is the relationship between a specialization and its implementation. It is an indication of the inheritance of behavior without the inheritance of structure.

Association: Association is represented by drawing a line between classes. Associations represent structural relationships between classes and can be named to facilitate model understanding. If two classes are associated, you can navigate from an object of one class to an object of the class.

Aggregation: Aggregation is a special kind of association in which one class represents as the larger class that consists of a smaller class. It has the meaning of “has-a” relationship.

Performance Evaluation

FEC Decoder
STATE CHART DIAGRAM
At any given time, on object is in a particular state. The UML state diagram captures this bit of reality. The symbol at the top of the figure represents the start state and symbol at the bottom represents the end state. State chart diagrams describe the behavior of an individual object as a number of states and transitions between these states. A state chart represents a particular set of values for an object. The sequence diagram focuses on the messages exchanged between objects, the state chart diagrams focuses on the transition between states.

REFERENCES


