October 2020

SELECTIVE FORWARD ERROR CORRECTION WITH FULL DUPLEX FEEDBACK LOOP

Pascal Thubert

Eric Levy-Abegnoli

Patrick Wetterwald

Jonas Zaddach

SELECTIVE FORWARD ERROR CORRECTION WITH FULL DUPLEX FEEDBACK LOOP

AUTHORS:
Pascal Thubert
Eric Levy- Abegnoli
Patrick Wetterwald
Jonas Zaddach

ABSTRACT

Numerous techniques exist for detecting and correcting errors that are introduced during the transmission of information (e.g., a data frame) between two pieces of network equipment, including, for example, forward error correction (FEC), cyclic redundancy check (CRC), hybrid automatic repeat request (HARQ), etc. Use of such techniques carries a cost, often shared between the transmitter and the receiver, comprising increased latency, consumption of bandwidth, the use of computational resources for verification and correction, etc. Techniques are presented herein that support a new method for detecting and correcting errors that leverages a return channel of a bidirectional radio environment to provide a feedback loop through which FEC may be focused just on the areas of a frame that are poorly received, thereby avoiding the latency, bandwidth, etc. costs that would be associated with retransmission of areas of the frame that are well received. The techniques presented herein build on new capabilities of full duplex radios and apply to, for example, Wi-Fi® 6 and 7 and Third Generation Partnership Project (3GPP) Fifth Generation (5G) networks.

DETAILED DESCRIPTION

There are current efforts to apply Time-Sensitive Networking (TSN) and Deterministic Networking (DetNet) to, for example, Wi-Fi® 6 and 7. Bringing determinism to a packet network means eliminating the statistical effects of multiplexing that result in probabilistic jitter and loss. Wireless networks operate on a shared medium where uncontrolled interference, including the self-induced multipath fading and objects passing in the Fresnel zone, may cause random transmission losses that necessitate error

recovery mechanisms, which adds new dimensions to the statistical effects that impact delivery. The problem at hand is to reduce these effects to the maximum extent possible. A typical use case is a control loop that consists of a unidirectional transmission sensor to a logic controller (e.g., a programmable logic controller (PLC)) and the PLC to an actuator. This frees the return channel of bidirectional radios free to exploit aspects of the techniques that are presented herein.

Multiple techniques exist to detect and correct errors introduced during the transmission of information (e.g., a data frame) between two pieces of network equipment. Detection is usually performed through some sort of computation (e.g., a hash, etc.) introduced into the frame by the transmitter and subsequently checked by the receiver.

Correction of an error may involve a full retransmission of the frame, a partial retransmission of the frame, or a correction of an error at the receiver (provided that the frame carries sufficient data redundancy to enable the receiver to do so).

One known technique, CRC, enables error detection (through the computation of a code based on the remainder of a polynomial division of the frame data) but does not support correction of the error. Acknowledgment and retry (e.g., ARQ) based on CRC is the most common error recovery method at the Media Access Control (MAC) layer.

Another known technique, FEC, allows a sender to insert redundant error-correcting data along with the data frame. Upon receiving the information, the recipient can use that extra data to check for errors and then correct them. The error correcting data can cover blocks (pieces) of the frame, of equal or variable length.

A synthesis of FEC and CRC, HARQ, involves the use of both FEC and ARQ at the same time.

Convolutional codes are used extensively to achieve reliable data transfer in numerous applications, such as digital video, radio, mobile communications (e.g., in Global System for Mobile Communications (GSM), General Packet Radio Service (GPRS), Enhanced Data rates for GSM Evolution (EDGE) and 3GPP Third Generation (3G) networks until 3GPP Release 7), and satellite communications. These codes are often implemented in combination with a hard decision code, particularly Reed–Solomon codes. Prior to turbo codes such constructions were the most efficient, coming closest to the Shannon limit.

The first class of turbo code was the parallel concatenated convolutional code (PCCC). Since the introduction of the original parallel turbo codes in 1993, many other classes of turbo code have been discovered, including serial versions of serial concatenated convolutional codes and repeat-accumulate codes. Iterative turbo decoding methods have also been applied to more conventional FEC systems, including Reed-Solomon corrected convolutional codes, although these systems are too complex for practical implementations of iterative decoders. Turbo equalization also flowed from the concept of turbo coding.

Mechanisms such as these come with a price, often shared between the transmitter and the receiver. Retransmission creates unbounded latency and is generally not acceptable in deterministic networks. FEC (including turbo and block codes) introduces a significant amount of redundancy in the frame and also carries a computational cost for verification and correction. The amount of redundancy is blindly selected in advance, so it may end up being too much, not enough, or misplaced.

Presented herein are techniques that support a new method for detecting and correcting errors that focusses the FEC on the areas of the frame that are poorly received. Such a method avoids the latency due to retransmission and the consumption of bandwidth associated with performing FEC on areas of the frame that are well received. It builds on new capabilities of full duplex radios and applies to Wi-Fi 6 and 7 and 3GPP 5G networks.

For example, with Wi-Fi 6 and 7 it has become possible for a station (STA) or an access point (AP) to send and receive a frame at the same time (during the same transmission operation). Such APs can be used, for example, for bridging and meshing. An example use case would have an industrial robot equipped with a (redundant) bridging AP that aggregates its traffic and enables mobility. Aspects of the techniques presented herein support among other things reliable transmission and avoid the latency that arises with ARQ and back-off, which may compromise the machine-to-machine (M2M) operation.

The techniques presented herein leverage elements of the Institute of Electrical and Electronics Engineers (IEEE) 802.11be standard and, in particular, in-band full duplex (IBFD) technology (a feature that will be available in 5G and Wi-Fi 7 APs of the second phase) that are able to acknowledge bits almost on the fly. When a frame is sent, additional

FEC is added to the end of the frame to ensure that it is received correctly the first time once the FEC is applied. That FEC is dynamic and depends upon what the receiver parsed.

The least latency is obtained by a physical layer (PHY) reflection, which causes the sender to correct errors that may have been injected in either direction. Aspects of the techniques presented herein support optimizations where a block-wise soft error value is returned and the amount of additional FEC depends on that error value, acting as a turbo to the encoder process.

Further, aspects of the techniques presented herein support error correction without explicit acknowledgement and retransmission (which creates latency that can be problematic) and without inserting redundant data into the frame (which comes with an overhead, both in terms of bandwidth and processing on the recipient side).

An initial use case may comprise one of two APs bridging traffic from a STA over Wi-Fi. A secondary use case may have a STA sending traffic to an AP. This second use case is more challenging, as it requires the STA to support simultaneous bidirectional transmission.

One aspect of the techniques presented herein involves a receiving side mirroring the data that it receives back toward a sending side as soon as it starts receiving. Since the transmitting side can receive and send at the same time, it compares the mirrored data with what it sent to search for differences (e.g., errors). The comparison can be performed byte for byte or block for block (e.g., with a precomputed cyclic redundancy check (CRC)).

As soon as an error is detected, a correction is inserted on the fly into the frame still being transmitted. The goal is to send the frame and the correction (if any) during the same transmission operation, to avoid latency. Further, the amount of redundancy in the FEC can be increased or decreased adaptively, based on the observed error rate.

The receiving side applies the correction on the fly as well, to end up with a correct frame.

Aspects of the techniques that are presented herein may be explicated with reference to Figure 1, below.

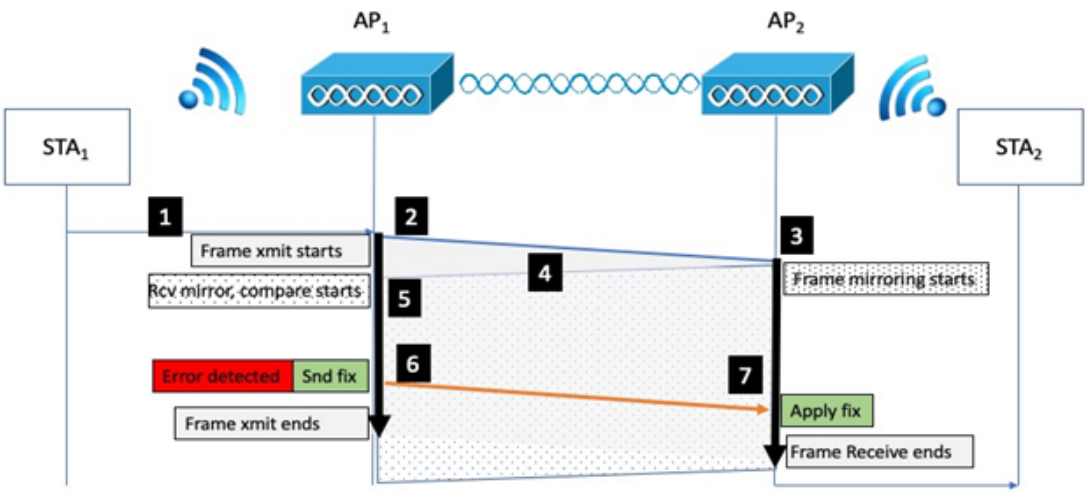

Figure 1: Error Correction with Feedback Loop

As illustrated in the above figure, various steps may include:

- Step 1, where STA_1 transmits a frame to AP_1.
- Step 2, where AP_1 obtains a transmit/receive slot and begins transmitting to AP_2.
- Step 3, where AP_2 begins receiving.
- Step 4, where AP_2 mirrors to AP_1 on the fly.
- Step 5, where AP_1 receives mirrored data and compares that data against what was sent. The comparison may be, for example, bit by bit or block by block.
- Step 6, where if AP_1 detects a difference (e.g., an error) it resends the correction, along with instructions for replacing the correction.
- Step 7, where AP_2 (which is still receiving the original frame) receives the correction and applies it. Note that the correction itself may also be mirrored (e.g., as a way of detecting excessive error volumes or rates).

Note that at Step 6 a sender (e.g., AP_1) may define a block size as part of the coding mechanism or outside of it. For an echo of a block that is not received correctly, the block may be resent as a binary difference (e.g., like a software patch). In the simplest form that would comprise N contiguous bits with offset information. Integrated with a turbo code, that would allow for the resending of one block and its FEC, indicating the offset of the

block, or just additional recovery bits that are sufficient, along with the partially corrupted block as captured in the echo transmission, to reconstruct the block correctly.

In general, the techniques that are presented herein improve upon a HARQ mechanism that is operating in full-duplex by sending acknowledgements faster to reduce latency. As a result, a first frame is always received correctly, as the acknowledgement (ACK) is done on the fly during the transmission with just one block latency, so the retransmission is literally appended to the transmission. Thus there is no need for another transmission operation (txop). This is important for DetNet or TSN traffic as ARQ and back-off may defeat the bounded latency requirement.

Aspects of the above, focusing on a sender's standpoint, may be examined with reference to Figure 2, below.

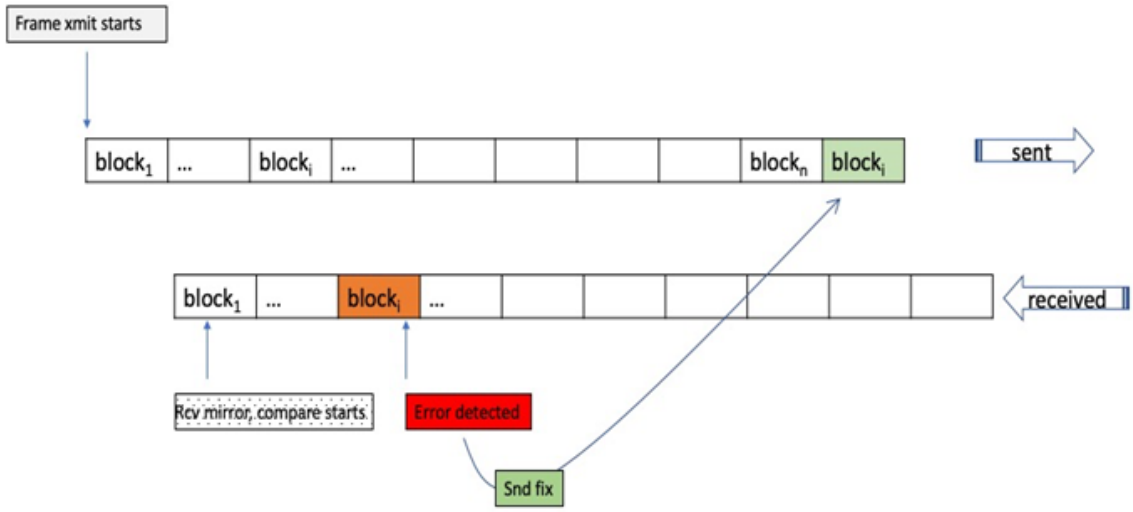

Figure 2: Sender Perspective

Similarly, aspects of the above, but now focusing on a receiver's standpoint, may be examined with reference to Figure 3, below.

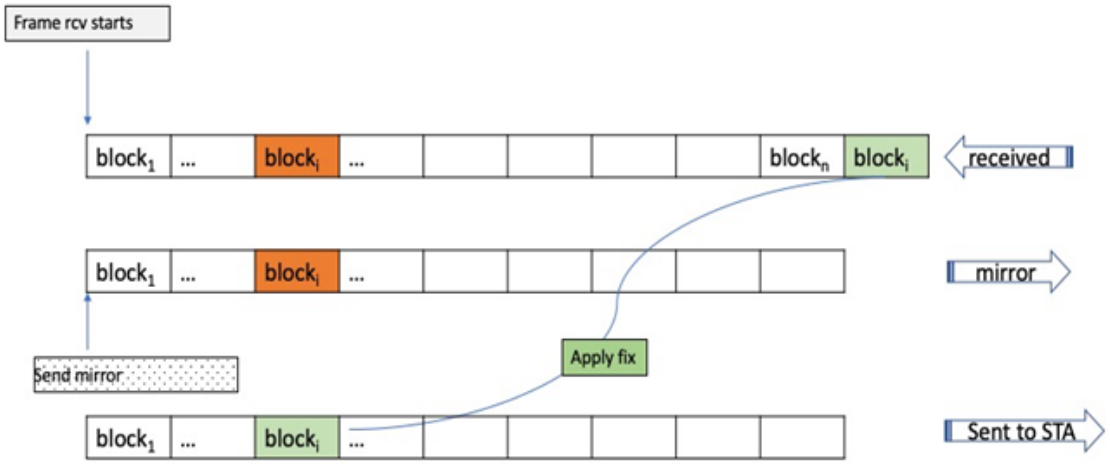

Figure 3: Receiver Perspective

The term "turbo code" arose from the feedback loop used during normal turbo code decoding, which was analogized to the exhaust feedback used for engine turbocharging. Hagenauer has argued the term turbo code is a misnomer since there is no feedback involved in the encoding process. Aspects of the techniques presented herein leverage the return channel of a bidirectional radio to provide a feedback loop, so that redundancy is only applied on blocks of data that are damaged after a round trip. It is important to note that it does not matter whether such damage occurred on the way out or on the way back.

One possible way in which the techniques presented herein may be employed is in conjunction with turbo codes where the data blocks are larger and the parity blocks are smaller. As a result a receiver has a softer decision on whether it received the data correctly (meaning lower certainty). It could then echo, in full, only the blocks that are uncertain (i.e., which fall below a certain threshold) and compress the returned data when the certainty is still very high. In such a way the techniques that are presented herein finally afford the term turbo code its original name.

Consider an environment comprising full-duplex transmission between two APs with reciprocity of wireless channels. In such an environment the probability of an error arising in a AP1 to AP2 direction is almost the same as the mirror path (i.e., from AP2 to AP1). Therefore, a case may arise where AP2 is receiving the blocks correctly but since

the error is introduced in the mirror link, AP1 needs to retransmit, resulting in unnecessary retransmissions.

Employing aspects of the techniques that are presented herein, the (turbo) code sends the frame as small blocks, each block with parity and shifted parity allowing for reconstruction of the misread bits. The techniques presented herein recognize the fraction of blocks that did not make the round trip. Those blocks are added at the end of the same frame as additional FEC. Note that a core concept is detection of the errors while transmitting and resending just the bad bits as a trailing information to the packet.

For example, consider a case where there is a 1/10 chance that a block is corrupted and that a frame is 10 blocks. There will be 1 block damaged on the way in, and one on the way back. The 2 blocks are discovered on the fly and re-appended at the end of the same frame, as an additional dynamic FEC -- one justified, the other due to the return path. While this makes the transmission of the frame 20% longer than without the techniques presented herein, importantly it avoids an ARQ which would mean a later 100% retransmission. Thus latency is improved, bandwidth is saved, etc.

One possible optimization to the approach discussed above involves sending (as a series of 'mini' acknowledgements) just the identifiers of the blocks that were received with certainty, in which case the error on the way back is discounted.

It is important to recognize that even if, for any number of reasons, it is not possible to immediately send a correction (e.g., Step 6 in Figure 1, above) the techniques that are presented herein are still valuable as it is desirable to have an early indication of errors (as opposed to waiting until completion).

It is also important to recognize that the techniques presented herein may operate without the use of a code (e.g., a turbo code). In such a case all the bits may be reflected as-is and a correcting difference block may be appended at the end of the same frame. If a code like a turbo code is used, the techniques that are presented herein allow for a cost reduction by making things 'softer.' For example, a decision regarding whether a block was correctly received or not may be less certain (perhaps employing a scale ranging between -127 and +127). Below a certain threshold a decision may be taken to resend. This may be accomplished through a per-block ACK/negative-acknowledgement (NACK) that is sent on the return channel instead of the original bits.

In the above discussion an error, in the most basic case, is a binary difference between an outgoing stream and a reflected stream, with a delay of a few bits between them. In such a case a binary patch may be generated on the fly and added at the end of the frame.

In the more interesting case of a PHY with a block wise coding system - e.g., turbo codes - an error is a soft recognition of the chances that a block was received below a threshold. This then is what is reflected back, per block. The sender adds additional FEC for the blocks that are below a threshold (e.g., by doing an additional shift that results in a third parity block) . Combined with what was received for the block, this gives a probability above the threshold at the receiver. An intermediate variation reflects all the blocks and their parity bits as they come, and the sender uses that incomplete block and parity to compute the minimal additional parity that will help restore the block with enough certainty.

For 5G variations, the resource block may be managed through techniques such as IEEE 802.15.4 Time-Slotted Channel Hopping (TSCH), with sufficient delay after the beginning of the resource block for the echo to absorb the round trip.

For blocks comprising a turbo code, the original message may be partitioned into blocks and each block then encoded twice, once directly and once through an Interleaver (as illustrated in Figure 4, below) that spreads the information over the coded block.

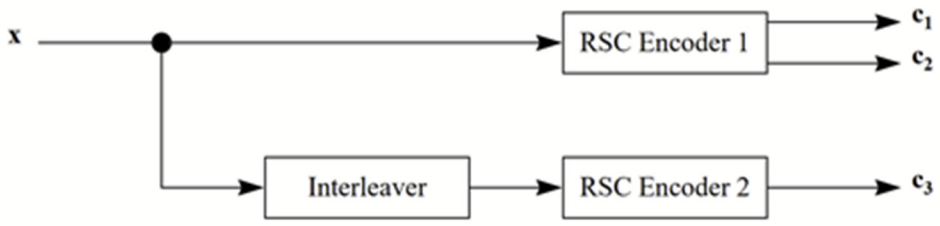

Figure 4: Duplicate Encoding with an Interleaver

The techniques that are presented herein provide a feedback loop from the receive end, which could be the original signal (e.g., a PHY reflection), the reconstructed signal (e.g., what was understood is resent and reencoded), or an acknowledgement per block indicating the likelihood that the block was received correctly. Accordingly, the sender decides what addition is needed for each block -- e.g., a complete block plus two codes, or one or two extra codes through extra interleaving but not the systematic data (the input),

etc. What may be resent could also be a binary patch, with a pair of codes to ensure that the patch is correctly received.

In summary, techniques have been presented that support a new method for detecting and correcting errors by leveraging a return channel of a bidirectional radio environment to provide a feedback loop through which FEC may be focused just on the areas of a frame that are poorly received, thereby avoiding the latency, bandwidth, etc. costs that would be associated with the retransmission of areas of the frame that are well received.