Punctured Turbo Code-Based ARQ Schemes in Rayleigh Fading

M. A. Kousa[1], **S. A. Al-Semari**[1] **and A. B. Adinoyi**[2], **A. H. Muqaibel**[3]

[1]Electrical Engineering Department, King Fahd University of Petroleum & Minerals, Saudi Arabia.

[2]Systems & Computer Engineering Department, Carleton University, Ottawa, ON Canada

[3] Electrical & Computer Engineering Department, Virginia Polytechnic Institute & State University, VA

Abstract

Turbo codes based automatic repeat request can be a powerful technique to enhance the throughput efficiency of data networks giving the astonishing performance of turbo codes. In wireless channels, the throughput can be maximized by adaptively matching the error correction capability of the code to the prevailing channel conditions. This paper investigates the throughput efficiency of hybrid ARQ based on punctured turbo. The throughput is enhanced when code sets are designed along the guidelines of good puncturing.

1 Introduction

Automatic repeat request (ARQ) is one of the most powerful techniques to enhance the reliability of transmission for data networks. A basic ARQ scheme is based on error detection and retransmission. In this basic scheme, the message to be transmitted is encoded using an error detection code. The decoder checks the received message, and if no error is detected the message is delivered to the sink. A positive acknowledgment (ACK) may be sent back to confirm the successful reception of the message. In the event that error is detected, the erroneous message is discarded and a negative acknowledgment (NACK) is sent to the transmitter requesting a retransmission of the same message. The process is repeated until the message is successfully received [1] .

The basic ARQ scheme described above is highly reliable but suffers from very low throughput efficiency when the channel quality degrades. To improve the throughput efficiency of ARQ schemes without sacrificing their reliability, hybrid ARQ/FEC schemes were proposed. Such schemes can be broadly classified as Type-I ARQ or Type-II ARQ. Type-I ARQ is a fixed-rate coding scheme that jointly employs one code for error correction and another code for error detection. It suites channels that are uniformly noisy but subject to infrequent impulses of noise. For time-varying channels, like those encountered in wireless transmissions, the throughput of the system can be maximized by adaptively matching the error correction capability of the code to the prevailing channel condition. This forms the motivation for type-II ARQ schemes. The problem of constructing adaptive-rate error correction codes suitable for Type-II ARQ schemes has attracted many researchers for decades.

For instance, an adaptive-rate scheme based on cascaded Hamming codes has been designed and analyzed in [2]. Hagenauer [3] proposed and analyzed a class of codes called Rate-Compatible Punctured Convolutional codes (RCPC), which are families of codes obtained through puncturing of a mother convolutional code. The astonishing performance of turbo codes [4] stimulated many researchers to consider their application in ARQ schemes. Recently, a capacity-approaching Hybrid ARQ techniques based on turbo codes were presented [5]. The authors have used a parity spreading interleaver to realize a flexible puncturing scheme. Then, the reordered parity bits are transmitted sequentially in increments of predetermined size that can ensure rate-compatible codes. In this paper, puncturing will be considered to adaptively match the error correction power of turbo code to the time-varying wireless channel. The puncturing patterns are properly designed to yield the best throughput performance.

The rest of the paper is organized as follows. In the next section the RCPT-ARQ scheme is described, this is followed by the simulation results and finally conclusions will be drawn.

2 The Channel Model & RCPT-ARQ Protocol Description

The RCPT encoder encodes the information packet and stores all the coded bits in buffers for any possible retransmissions. The rate selector determines which parity bits are to be transmitted to ensure the codes adaptation to the state of the channel. The Rayleigh flat fading channel is assumed. The fade samples are zero mean and unit variance complex Gaussian variables. It is assumed that error-free (reliable), low capacity feedback channel does exist over which the receiving system can transmit the acknowledgements.

A family of rate compatible punctured turbo codes (RCPT) can be obtained by puncturing a rate 1/n-mother turbo code. A set of puncturing rules is defined in the puncturing pattern. Examples of such patterns are shown in Table 2- 4 for different periods. Table 2 shows three sets of puncturing patterns for period $p = 2$, R_{20}, R_{2cc} and R_{21}. The code sets R_{20} and R_{2cc} produce the code rates $1, 1/2, 1/3$ while the set, R_{21} produces the code rates $1, 2/3, 1/2, 2/5, 1/3$. Note that in R_{2cc} all bits corresponding to a parity branch have been punctured. This pattern therefore, transforms the turbo code into a convolutional code. It is important to mention that rate compatibility is required for this Hybrid ARQ schemes. This restricts puncturing rule to such a form that all of the code symbols of high-rate punctured code are embedded in the lower rate codes. At each step the transmitter needs only to transmit supplemental code symbols, reason why type-II ARQ scheme is often called incremental redundancy.

Considering puncturing patterns of period 4, one can produce the rates, $\{1, \frac{4}{5}, \frac{2}{3}, \frac{4}{7}, \frac{1}{2}, \frac{4}{9}, \frac{2}{5}, \frac{4}{11}, \frac{1}{3}\}$. An optimistic assumption of a less pathological channel may try to use all codes in this set. Hence, high throughput may be obtained. However, we have made a rather pessimistic assumption of bad channels, and therefore considered an exponential jump in the code rate selection. This gives rise to few code rates in the set we have used in the retransmission, for example, for period, $p = 4$ puncturing pattern R_{41}, R_{42} and R_{43} shown in the Table 3. Along similar lines, four sets have been considered for period 8, and these are shown in Table 4 [1]. The RCPT-ARQ protocol studied will now be described.

[1]Note that the entries in the table are the decimal representation of the bits forming the puncturing pattern, partitioned starting from right, e.g 6=110, 1=001, etc

ARQ systems can be implemented by choosing RCPT codes in a strategic way for each retransmission. The RCPT codes can be implemented with stop-and-go, go-back-N, selective-repeat schemes. However, selective-repeat ARQ strategy is considered for its bandwidth efficiency and the ease at which it can be adapted to other schemes [6]. For simplicity of analysis, we have considered an infinite buffer size. The general steps involved in the RCPT-ARQ protocol are given as follows.

1) Encode the k^{th} packet with the rate 1/3 turbo code and buffer all the coded bits. Choose a set of puncturing patterns. Let m denote the set size.

2) First transmit the bits specified in the first pattern of the set (usually the information bits only).

3) The receiver assembles the received bits, inserting erasures for the bits that have not been transmitted according to the puncturing, and decoding is attempted. We have assumed that all error patterns can be detected. If the frame received by the sink is error free, an ACK is sent to the transmitter and the protocol is reset and returned to step 1. Otherwise, a NACK is transmitted back.

4) Upon reception of NACK the transmitter sends additional bits according to the next pattern of the set, then back to step 3.

The first m transmissions will all be distinct. They exhaust all the code rates in the set, reaching the full turbo code of rate 1/3. We say this scheme has a degree of freedom of adaptation of m. If an error-free packet cannot be passed yet, different design options can be considered. The simplest is to accept a given frame error rate and pass the frame as is. Another option is to hold the decoder in full turbo but the transmitter is reset to step 2. In this case popular code combining is done.

3 The Simulated Throughput Results

The performance of ARQ schemes is measured by its throughput. The throughput simulation results are presented for turbo-hybrid ARQ scheme for different parameter settings. First, we considered the effect of period of puncturing pattern on the throughput performance. We also considered for each puncturing period the effect of different puncturing patterns. Furthermore, we considered the situation where more steps are introduced in the puncturing rules.

We start by examining the effect of the selection of the puncturing patterns. For that purpose the period of

Table 1: Different puncturing patterns of period 4

$$P_{41} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}, \mathrm{P}_{42} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\text{and } P_{43} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

puncturing p and the size of the code set (adaptation stages) are kept fixed. We consider two code set sizes, $m = 3, 5$ and the puncturing periods, $p = 2, 4, 8$.

Figure 1 presents the throughput performance curves for schemes using puncturing patterns of period 4 from Table 1. For all curves, the puncturing patterns for the first and last stages are the same : only information bits, and the full code, respectively. For the intermediate stage, the middle (third) puncturing pattern of each of P_{41}, P_{42} and P_{43} shown in Table 1 is used. It is clearly seen that the performance obtained with P_{43} is woeful in comparison with others. The results show that the gain in throughput due to selecting a good puncturing pattern could reach more than 15 %.

Next, the throughput performance evaluation is extended to $m = 5$ (five steps of adaptation). It is worth noting that for higher periods additional flexibility is obtained in the number of intermediate steps before reaching the full turbo code. This implies that more degree of freedom is allowed in the puncturing rule. In fact, this can yield a better code distance properties that can translate into better code performance compared to the case of shorter puncturing periods. Figure 2 shows the effect of the position of the punctured bits for period 4. It is seen that code rate set of R_{41} gives a better performance than R_{42} or R_{43}. We have generally observed that for the code structure used, the bits retained or punctured of the parity sequences have direct effect on the code performance contrary to the opinion contained in [8]. Their conclusion may be based on the context of their investigation.

Next, we investigate the effect of the puncturing period on the code performance. For that reason we have m fixed, and we use the best pattern for each case. The value $m = 5$ is considered in Figure 3 for the best performance of periods 2, 4 and 8. We observed the improvement in the throughput by using a larger period. However, the improvement is of diminishing nature. The improvement of going from $p = 4$ to $p = 8$ is less than that from $p = 2$ to $p = 4$. This can be attributed to the freedom offered by larger period in distributing the puncturing bits.

Finally, we study the effect of the adaptation steps on the throughput performance. For this case we have compared the code rate set $\{1, 2/3, 1/2, 2/5, 1/3\}$ with the set $\{1, 1/2, 1/3\}$ for $p = 8$ in Figure 4. Generally, the higher the degree of freedom in selecting the code rate the higher the throughput performance. It was observed that the gain is more pronounced for larger periods than for shorter ones, an improvement of 10% is obtained due to increasing the number of adaptation steps for $p = 4, 8$.

In conclusion, the throughput efficiency of punctured turbo code-based ARQ scheme has been presented. If the puncturing is done along the guidelines provided in this paper the system throughput is substantially improved.

References

[1] S. Lin, D. Costello,and M. Miller, "Automatic repeat request error control schemes" , *IEEE Transactions on Communication*, VOL22, pp.5-17, Dec 1984.

[2] M. Kousa, and M. Rahman, "An Adaptive error control system using hybrid ARQ schemes" , *IEEE Transactions on Communications*, COM-39, no. 7, pp. 1049-57, July 1991

[3] J. Hagenauer, "Rate compatible punctured convolutional codes (RCPC-codes) and their application" , *IEEE Transactions on Communications*, Apr. 1988, vol. 36, pp. 389-400.

[4] Claude Berrou and Alain Glavieux "Near Optimum Error Correcting Coding And Decoding: Turbo-Codes". *IEEE Trans on Information Theory*, VOL-44, pp.1261-1271, October, 1996.

[5] R. Mantha and F. R. Kschischang, "A Capacity-Approaching Hybrid ARQ Scheme Using Turbo Codes" , *Global Telecommunications Conference*, 1999. GLOBECOM '99. pg. 2341 -2345.

[6] Douglas N. Rowitch and Laurence B. Milstein "On the Performance of Hybrid FEC/ARQ Systems Using Rate Compatible Punctured Turbo (RCPT) Codes". *IEEE Transactions on Communication*, VOL-48, pp.948-959, June, 2000.

Top

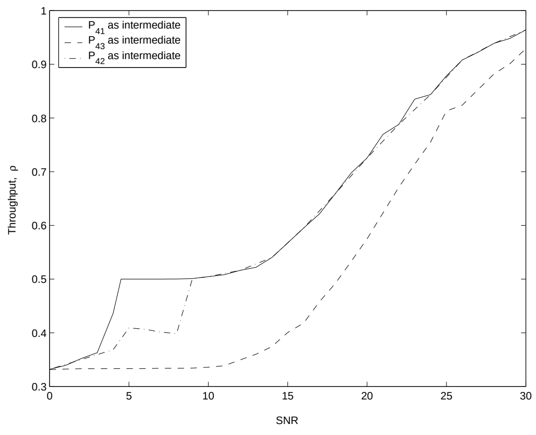

Figure 1: Throughput comparison of different patterns for $p = 4, m = 3$

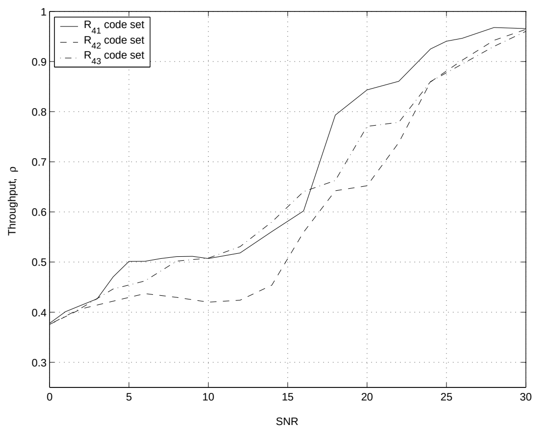

Figure 2: Throughput comparison of different patterns for $p = 4, m = 5$.

[7] Maan Kousa, Saud Al-Semari and Abdulkareem Adinoyi " Puntured Turbo Codes for Hybrid ARQ Schemes in Rayleigh Fading", *A report submitted to Graduate College, KFUPM*, 2002

[8] Fan Mo, S.C. Kwatra and Junghwan Kim "Analysis of puncturing pattern for High Rate Turbo Codes", *IEEE Military communication conference '99*, Vol. 1, Nov. 1999. pp.547-550.

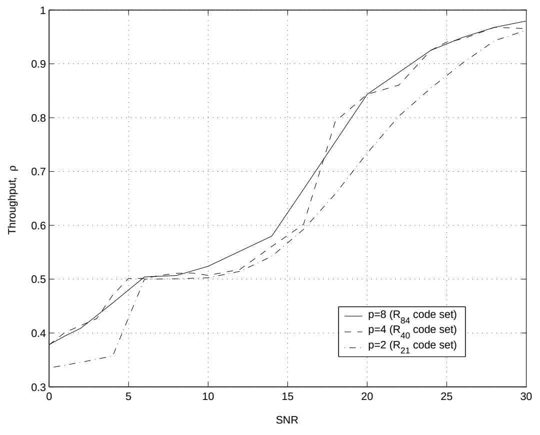

Figure 3: Throughput comparison of different periods with $m = 5$.

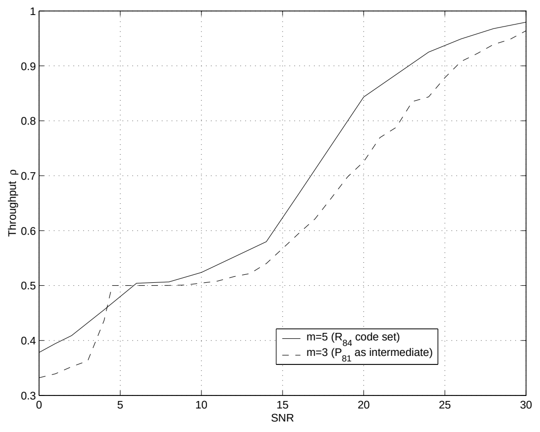

Figure 4: Throughput comparison different intermediate rates of periods 8.

Top

Table 2: Rate set puncturing pattern for period 2

Name	Pattern
R_{20}	$\begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$
R_{2cc}	$\begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$
R_{21}	$\begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$

Table 3: Rate set puncturing pattern for period 4

Name	Pattern
R_{41}	$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$
R_{42}	$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$
R_{43}	$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

Table 4: Rate set puncturing pattern for period 8

Name	Pattern
R_{81}	$\begin{bmatrix} 3 & 7 & 7 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 1 & 0 & 2 \\ 2 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 3 & 1 & 2 \\ 3 & 2 & 1 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 3 & 5 & 3 \\ 3 & 6 & 5 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 3 & 7 & 7 \\ 3 & 7 & 7 \end{bmatrix}$
R_{82}	$\begin{bmatrix} 3 & 7 & 7 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 2 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 2 & 5 & 2 \\ 1 & 2 & 5 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 3 & 7 & 2 \\ 1 & 3 & 7 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 3 & 7 & 7 \\ 3 & 7 & 7 \end{bmatrix}$
R_{83}	$\begin{bmatrix} 3 & 7 & 7 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 2 & 1 & 0 \\ 0 & 4 & 1 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 2 & 5 & 1 \\ 2 & 5 & 1 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 3 & 5 & 5 \\ 2 & 7 & 3 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 3 & 7 & 7 \\ 3 & 7 & 7 \end{bmatrix}$
R_{84}	$\begin{bmatrix} 3 & 7 & 7 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 1 & 0 & 4 \\ 0 & 2 & 1 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 1 & 4 & 6 \\ 2 & 3 & 1 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 3 & 5 & 6 \\ 2 & 7 & 3 \end{bmatrix}, \begin{bmatrix} 3 & 7 & 7 \\ 3 & 7 & 7 \\ 3 & 7 & 7 \end{bmatrix}$

Top