TSG-RAN Working Group1 meeting #3
Nynäshamn 22-26, March 1999

**TSGR1#3(99)189**

**Agenda Item:**

**Source:** Lucent Technologies

**Title:** A unifying code proposal for all data rates, block sizes and Quality of Service: Performance/Complexity trade-off

**Document for:** Discussion

---

## Introduction

In this document, we present a proposal for a unique solution to the channel coding required for UMTS applications, encompassing all data rates, block sizes, and quality of services. It is based on the serial concatenation of two 4-state convolutional codes with an interleaver in between.

Extended comparisons of the 4-state serially concatenated convolutional code (4-SCCC) with the 8-state parallel concatenated convolutional code (8-PCCC) are presented. The comparison encompasses all code block sizes, from 80 to 5120, and refers to three different quality of services, namely a bit error probability of $10^{-6}$, $10^{-7}$ and $10^{-8}$. For voice applications, and thus block sizes of 80 and 160, we also compare the performance of 4-SCCC with that of the 256-state convolutional code, which is suggested as working assumption for low data rate services.

The results have been obtained assuming an additive white Gaussian noise (AWGN) channel. We have three valid reasons for that: first, it would not have been possible to obtain such a large amount of results, down to $10^{-8}$, on a channel more complex to simulate. Second, the simulations over the correlated Phase 2 fading channel are not sufficiently reliable, because of the impossibility to consider a sufficient number of independent fading events in the simulations. Third, the hierarchy of performance obtained on an AWGN channel is maintained over the worse fading channel, with an obvious performance degradation which can be safely assumed to be uniform for the two coding schemes.

The results that will be presented later clearly show that the 4-SCCC solution offers very good performance over the whole range of block sizes and quality of services, and thus it should be used as a unique, simple code for all applications.

## Types of services, delays, data rates and block sizes

In the following Table 1, we list different type of services, and, for each category, the range of delay constraints, as coming from the transmission time intervals specified in [1], the information data rates, the information block sizes obtained as the product of the delay times the data rate, and, finally, the pertinent quality of services, as derived from [2].

*Table 1. Services, delays, data rates, block sizes, and QoS.*

| Services | Delay [ms] | Information data rates [kbit/s] | Maximum information block sizes for channel coding [bit] | QoS [BER, FER] |
|---|---|---|---|---|
| **Voice & LDD (Class A)** | 20 | 8 | 160 | [BER]$10^{-3}$-$10^{-5}$ |
| **Real –time data (Voice-band data, fax, video etc.) (Class B )** | 10-80 | 9.6<br>14.4<br>28.8<br>64<br>128 and beyond | 96-768<br>144-1152<br>288-2304<br>640-4320<br>1280-10240 and beyond | [BER]<$10^{-6}$ |
| **Non real time data: (packet data services like Internet services etc.) (Class C & D)** | 10-80 | 64<br>144<br>384 and beyond | 640-4320<br>1440-11520<br>1280 and beyond | [BER] <$10^{-8}$<br>[FER] ~$10^{-5}$<br>for FEC<br>[FER] =$10^{-2}$-$10^{-3}$<br>for ARQ |

Table 1 shows that the largest block size can amount to 11,520 bits and beyond. However, since the decoder complexity increases linearly with the block size, whereas the performance improvement saturates for block sizes around 1,000-2,000 (see next Figure 11 and Figure 12), it is convenient to limit the block size to 1,280-2,560.

## The results

We present two sets of results: the first refers to the computed true minimum (free) distance of 4-SCCC and 8-PCCC, and the second reports the outcomes of extensive simulation runs.

#### *The minimum (free) distance of the codes*

In Table 2, we show the minimum distances measured through a very fast, proprietary algorithm that tries as input information to the encoder all sequences with weight up to the minimum distance. They refer to 4-state SCCC and 8-state PCCC for all block sizes from 80 to 5,120, as the two reasonable extremes coming from Table 1. For the larger block sizes, the algorithm has not been able to measure the actual minimum distance. In those cases, there is an inequality in the corresponding table element. Its meaning is that we have found an error event with a weight equal to the upper bound of the inequality, and we have not found error events with a weight equal to the lower bound of the inequality, which was the largest size that our algorithm could manage.

The interleavers that have been used for the codes are the most recent interleavers proposed jointly by HNS and Nortel in Document [3] for 8-state PCCC, and interleavers kindly provided by NTT-DoCoMo for 4-state SCCC.

*Table 2 : Free distance of 4-state SCCC and 8-state PCCC for different interleaver types and sizes*

| | | HNS | Mix | MIL | Nortel |
|---|---|---|---|---|---|
| 80 | 4 SCCC | | | 18/1/6 | 23/1/2 |
| | 8 PCCC | | 11/1/1 | | |
| 16 | 4 SCCC | | | 23/1/6 | 28/10/56 |
| | 8 PCCC | | 17/2/6 | | |
| 32 | 4 SCCC | 29<d<32 | | 25/1/5 | 29/1 |
| | 8 PCCC | | 19/1/1 | 19/1/1 | 17/1/3 |
| 64 | 4 SCCC | 29<d <36 | | 29/2/18 | |
| | 8 PCCC | 18/1/2 | 19/1/1 | 19/1/3 | |
| 12 | 4 SCCC | | | 29<d<=32/1 | |
| | 8 PCCC | | 22/2/4 | d<=28/20 | |
| 25 | 4 SCCC | | | 29<d<=39/449 | |
| | 8 PCCC | | 23/1/1 | d<=26/1 | |
| 38 | 4 SCCC | | | | |
| | 8 PCCC | | d<26/4 | d<=25/1 | |
| 51 | 4 SCCC | Spread <62/1 | | 29<d<=54/1250 | |
| | 8 PCCC | | d<28/314 | d<=28/12 | |

Table 2 clearly shows that, for all block sizes, 4-state SCCC yields a significantly larger minimum distance than 8-state PCCC. Considering that the free distance of the rate 1/3, 256-state convolutional code is 18, 4-SCCC is also superior to 256-state convolutional code. In terms of performance, this means that 4-SCCC will have a lower error floor than 8-PCCC, and, also, that it should be less sensitive to the puncturing required for rate matching.

***Simulation results***

In the following we report the results of extensive simulation results for 4-state SCCC and 8-state PCCC for the following cases:

- Figure 1 and Figure 2: FER with a maximum of 10 iterations of the decoding algorithm
- Figure 3 and Figure 4: BER with a maximum of 10 iterations of the decoding algorithm
- Figure 5 and Figure 6: : FER with a maximum of 100 iterations of the decoding algorithm

- Figure 7 and Figure 8: : BER with a maximum of 100 iterations of the decoding algorithm

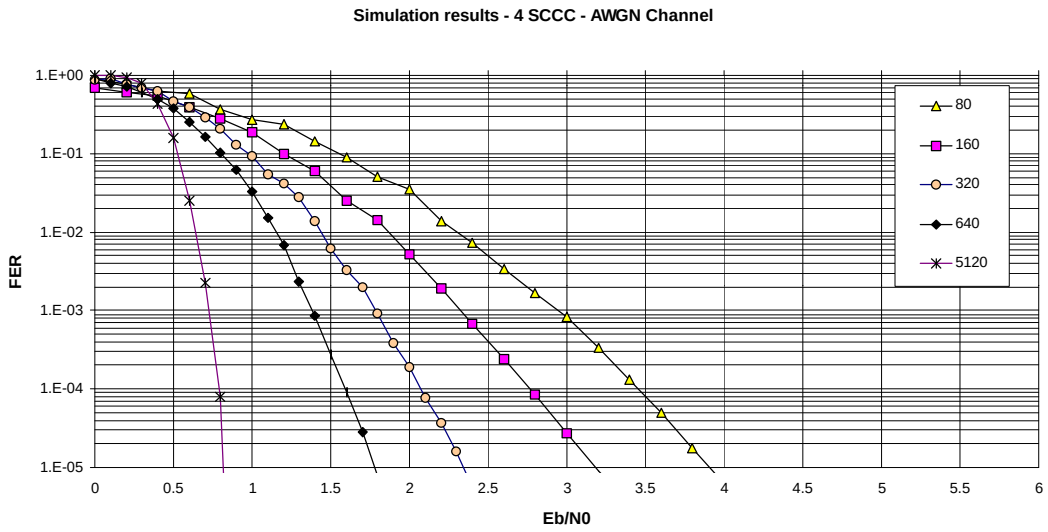


*Figure 1: Frame error probability of the 4-state SCCC over the AWGN channel with a maximum of 10 iterations. Information block sizes: 80, 160, 320, 640, and 5120.*

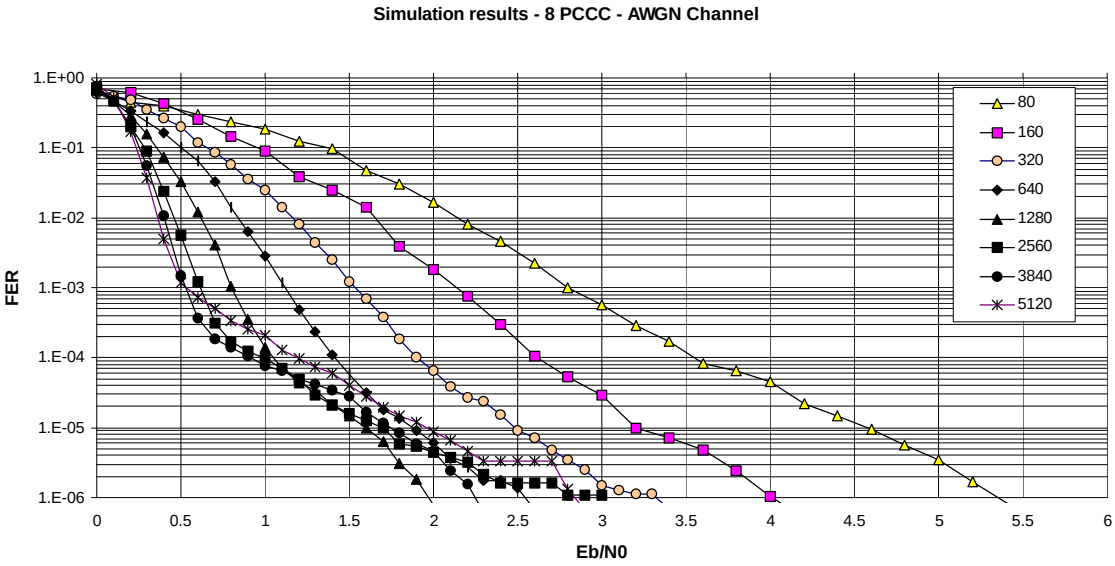


*Figure 2 Frame error probability of the 8-state PCCC over the AWGN channel with a maximum of 10 iterations. Information block sizes: 80, 160, 320, 640, 1280, 2560, 3840, and 5120.*

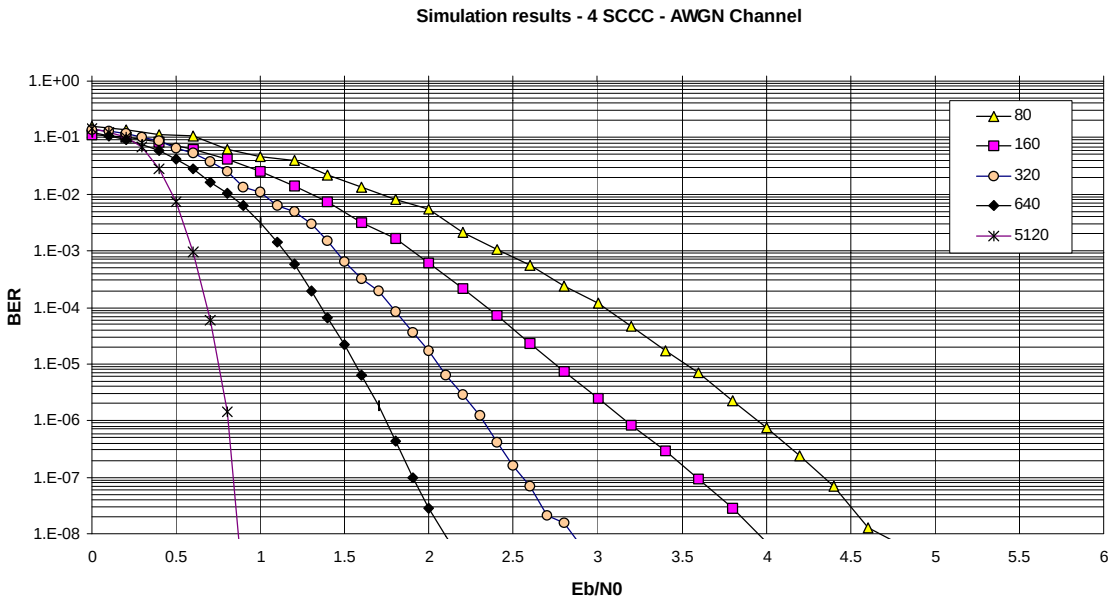


*Figure 3: Bit error probability of the 4-state SCCC over the AWGN channel with a maximum of 10 iterations. Information block sizes: 80, 160, 320, 640, and 5120.*

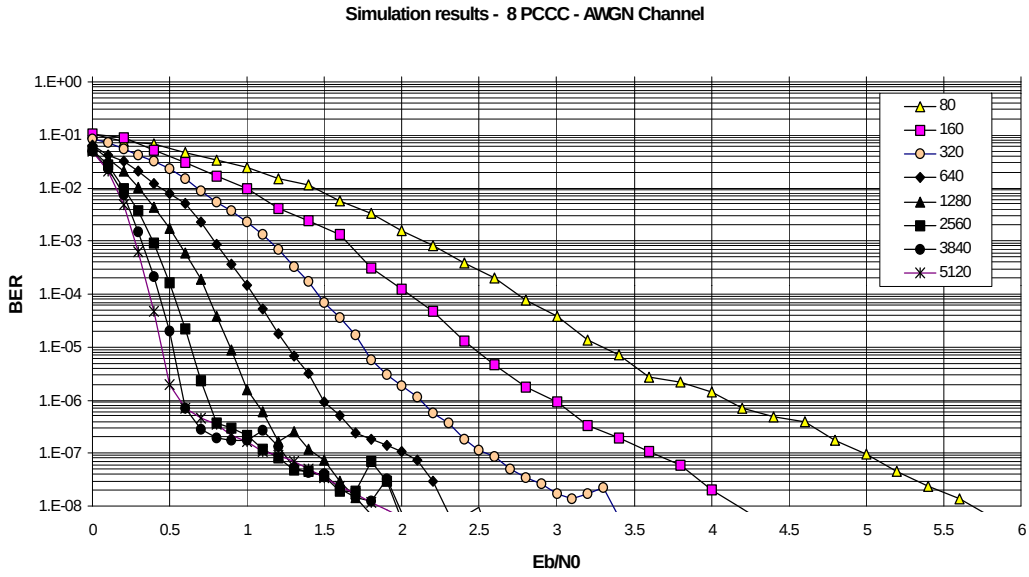


*Figure 4: Bit error probability of the 8-state PCCC over the AWGN channel with a maximum of 10 iterations. Information block sizes: 40, 80, 160, 320, 640, 1280, 2560, 3840, and 5120.*

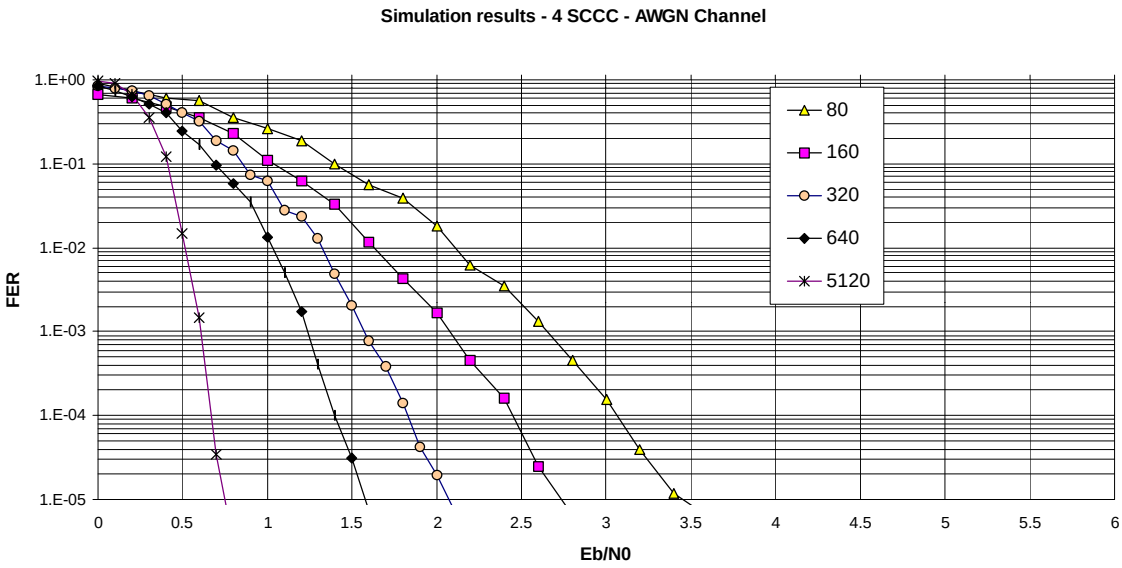


*Figure 5: Frame error probability of the 4-state SCCC over the AWGN channel with a maximum of 100 iterations. Information block sizes: 40, 80, 160, 320, 640, and 5120.*

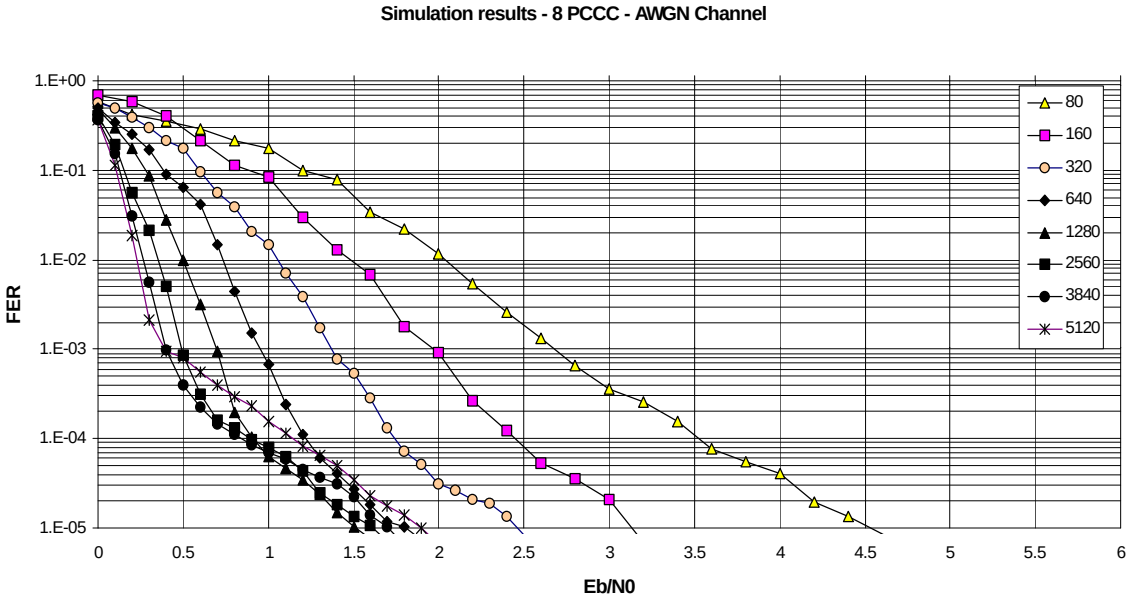


*Figure 6: Frame error probability of the 8-state PCCC over the AWGN channel with a maximum of 100 iterations. Information block sizes: 80, 160, 320, 640, 1280, 2560, 3840, and 5120.*

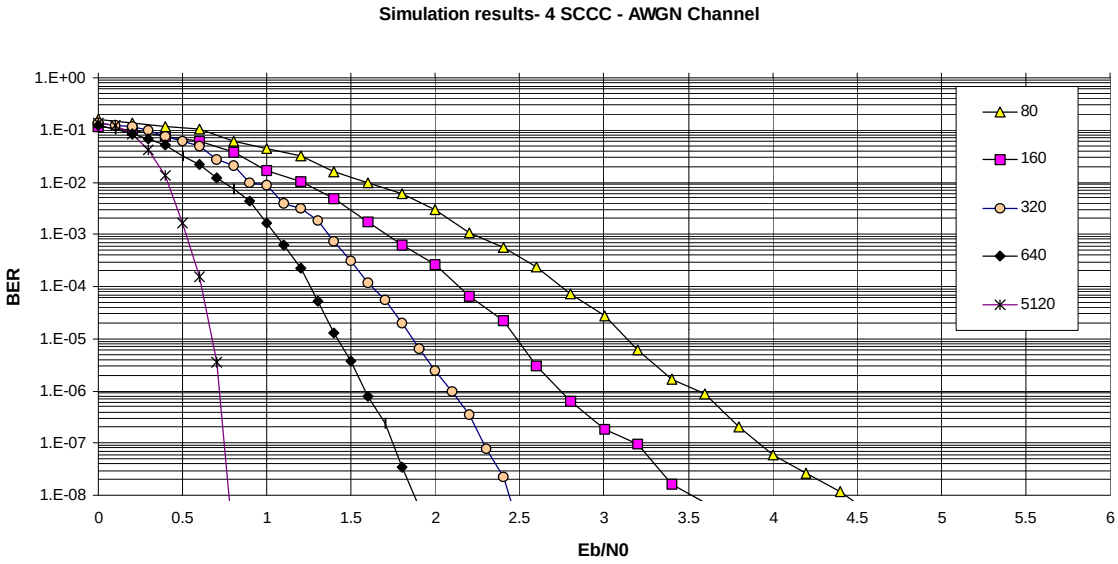


*Figure 7: Bit error probability of the 4-state SCCC over the AWGN channel with a maximum of 100 iterations. Information block sizes: 80, 160, 320, 640, and 5120.*

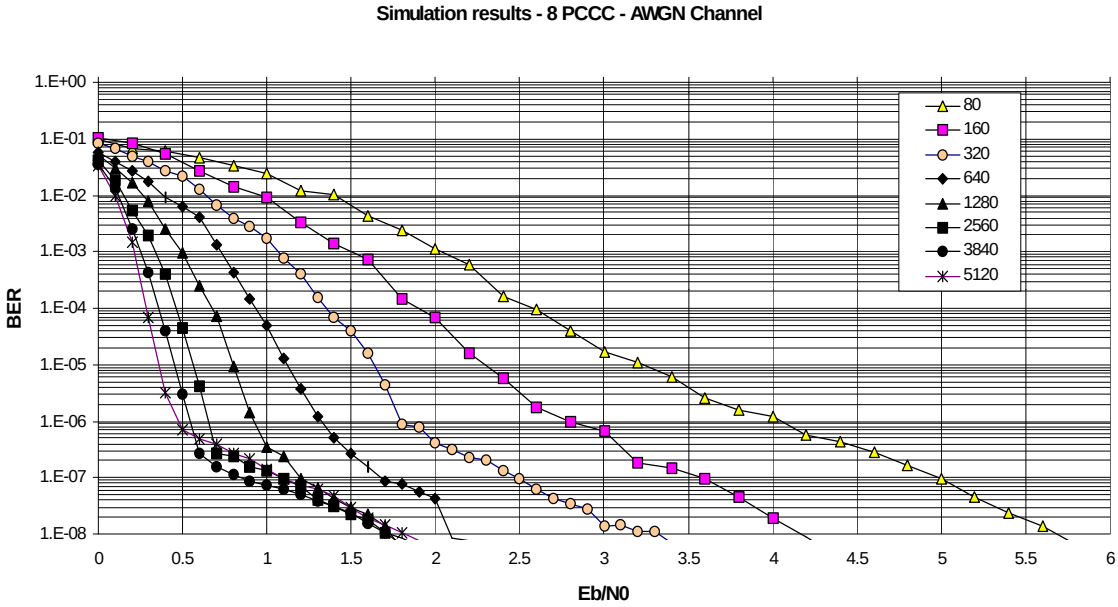


*Figure 8: Bit error probability of the 8-state PCCC over the AWGN channel with a maximum of 100 iterations. Information block sizes: 80, 160, 320, 640, 1280, 2560, 3840, and 5120.*

From the figures we can clearly appreciate the different behaviour of the two codes. In fact, 8-PCCC shows an evident error floor below $10^{-6}$ (BER) and $10^{-3}$ (FER).

***The required $E_b/N_0$ and implementation complexity versus block size***

From the previous simulation results, we have extracted, for different quality of services, namely a bit error probability of $10^{-6}$, $10^{-7}$ and $10^{-8}$ (as required from Table 1) and frame error probabilities of $10^{-1}$, $10^{-2}$, $10^{-3}$ and $10^{-5}$, the curves of the required $E_b/N_0$ versus block size for the two coding schemes. In the same curve, we also report a measure of the ASIC implementation complexity for the 4-SCCC and the 8-PCCC coding schemes, in terms of the required ASIC area in squared mm for a 0.5 μm CMOS technology, based on the results presented in [4].

Figure 9 shows the frame error probability (on the LHS ordinate axis) and implementation complexity (on the RHS ordinate axis) versus block size for the two coding schemes for a maximum of 10 iterations. Figure 10 gives the same informations for a maximum of 100 iterations.

Figure 11 and Figure 12 give the same kind of informations referring for the bit error probability.

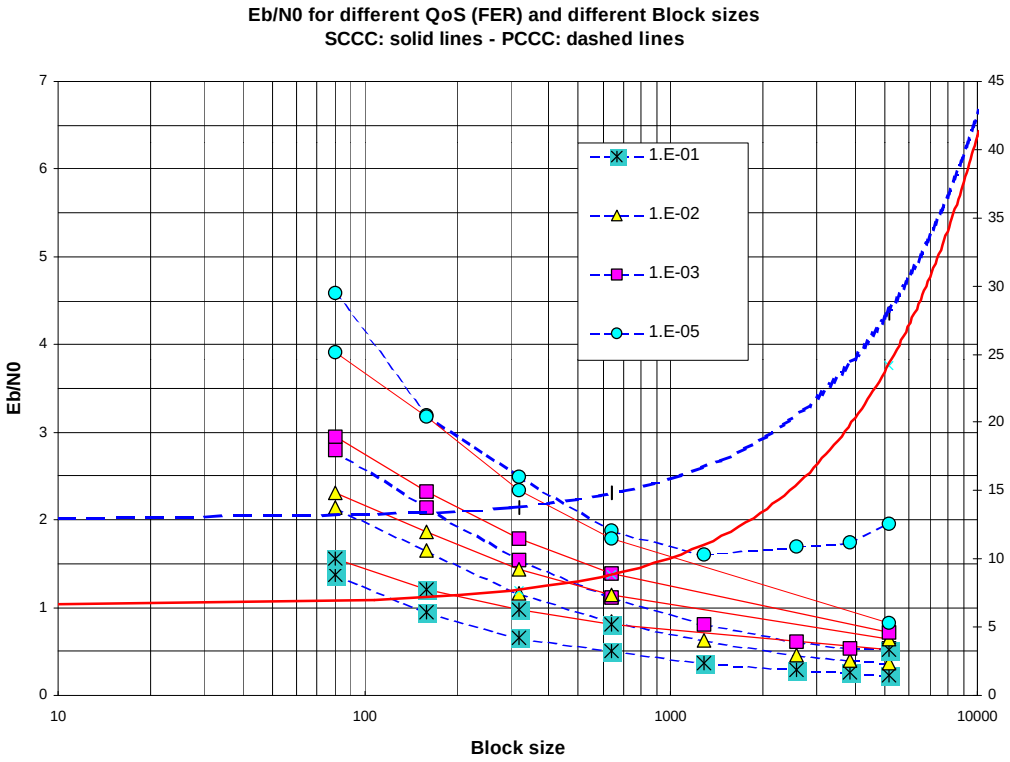


*Figure 9: Eb/N0 for different QoS (FER) and different Block sizes with a maximum of 10 iterations. The RHS ordinate reports the ASIC implementation complexity in square millimetres.*

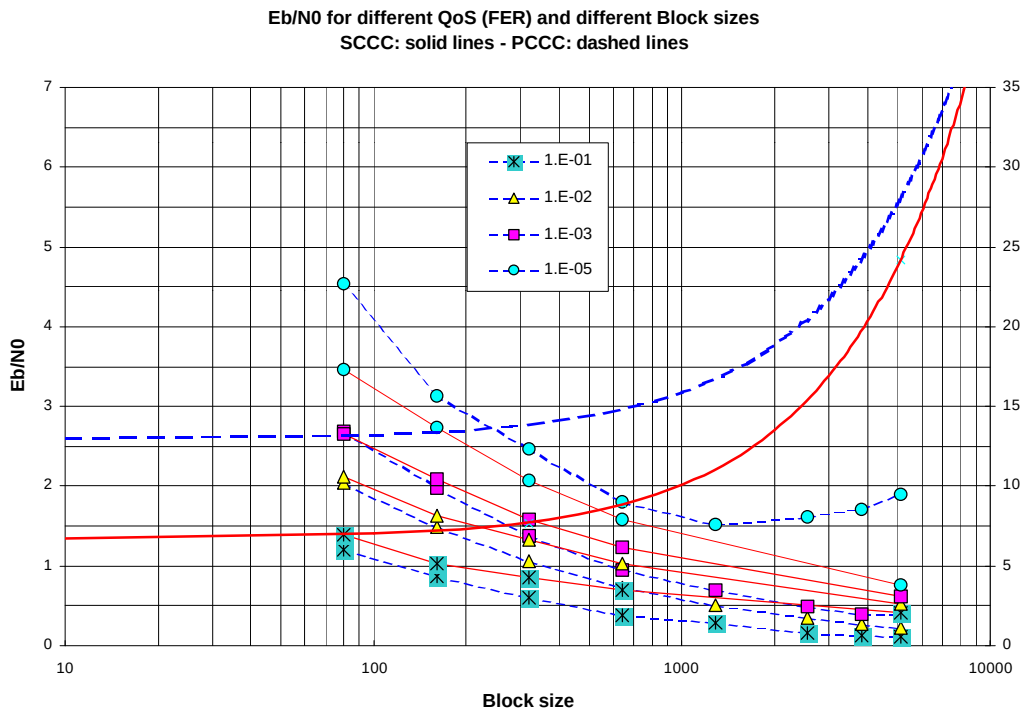


*Figure 10: Eb/N0 for different QoS (FER) and different Block sizes with a maximum of 100 iterations. The RHS ordinate reports the ASIC implementation complexity in square millimetres.*

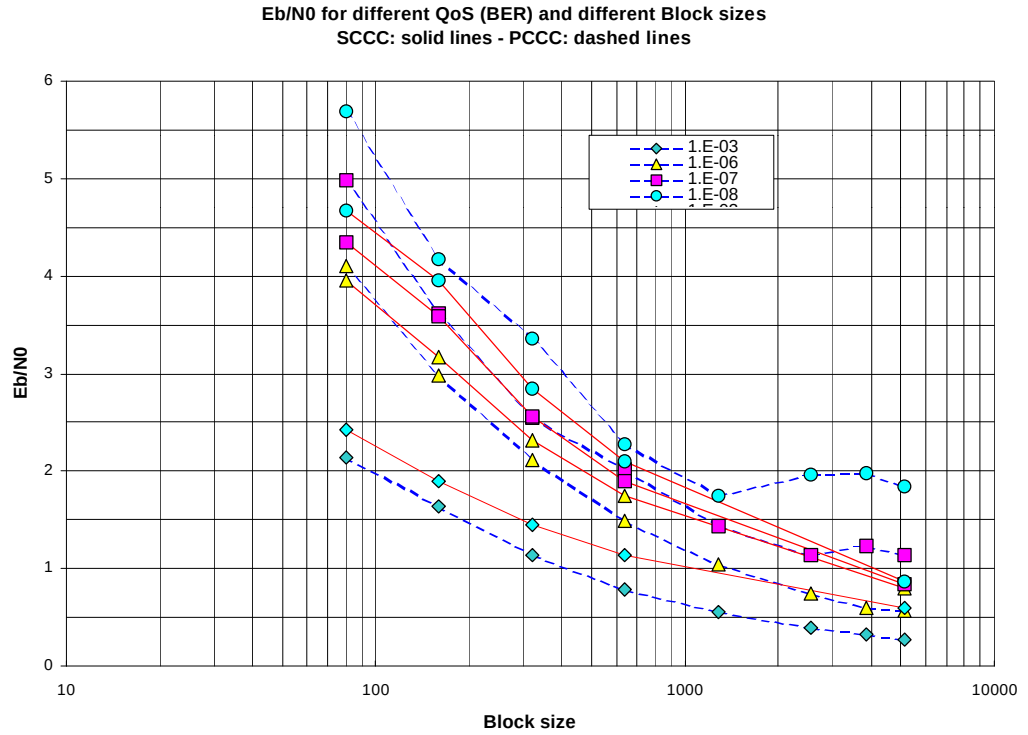


*Figure 11: Eb/N0 for different QoS (BER) and different Block sizes with a maximum of 10 iterations.*

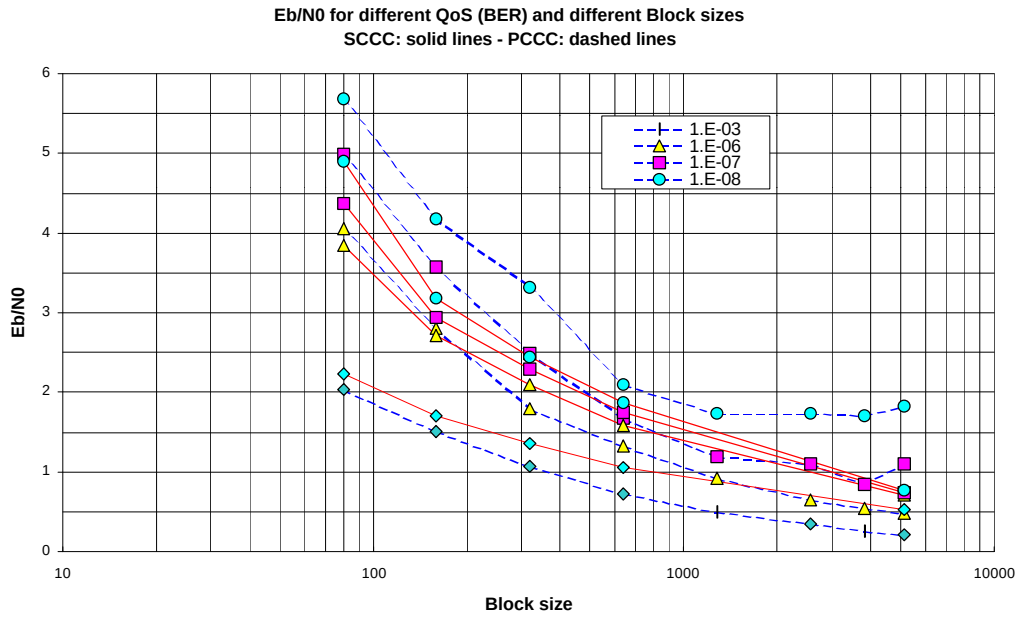


*Figure 12: Eb/N0 for different QoS (BER) and different Block sizes with a maximum of 100 iterations.*

*Table 3: Comparison in Eb/N0, capacity and complexity gains of 4 SCCC versus 8 PCCC for all block sizes, high quality services (BER). A maximum of 10 iterations of the decoding algorithm.*

| BER | 1.00E-06 | | 1.00E-07 | | 1.00E-08 | | |
|---|---|---|---|---|---|---|---|
| Block size | Gain in Eb/N0 [dB] | Gain in Capacity | Gain in Eb/N0 [dB] | Gain in Capacity | Gain in Eb/N0 [dB] | Gain in Capacity | Complexity gain |
| ***80*** | 0.16 | 3.62% | 0.64 | 13.70% | 1.01 | 20.75% | 90% |
| ***160*** | -0.19 | -4.47% | 0.03 | 0.69% | 0.22 | 4.94% | 85% |
| ***320*** | -0.20 | -4.71% | -0.02 | -0.46% | 0.52 | 11.28% | 80% |
| ***640*** | -0.25 | -5.93% | 0.11 | 2.50% | 0.18 | 4.06% | 68% |
| ***5120*** | -0.24 | -5.68% | 0.31 | 6.89% | 0.98 | 20.20% | 16% |

*Table 4: Comparison in Eb/N0, capacity and complexity gains of 4 SCCC versus 8 PCCC for all block sizes, high quality services (FER). A maximum of 10 iterations of the decoding algorithm.*

| FER | 1.00E-03 | | 1.00E-04 | | 1.00E-05 | | Complexity gain |
|---|---|---|---|---|---|---|---|
| Block size | Gain in Eb/N0 [dB] | Gain in Capacity | Gain in Eb/N0 [dB] | Gain in Capacity | Gain in Eb/N0 [dB] | Gain in Capacity | |
| **80** | -0.14 | -3.28% | 0.11 | 2.50% | 0.67 | 14.30% | 90% |
| **160** | -0.19 | -4.47% | -0.19 | -4.47% | 0.02 | 0.46% | 85% |
| **320** | -0.25 | -5.93% | -0.18 | -4.23% | 0.14 | 3.17% | 80% |
| **640** | -0.27 | -6.41% | -0.20 | -4.71% | 0.10 | 2.28% | 68% |
| **5120** | -0.19 | -4.47% | 0.43 | 9.43% | 1.14 | 23.09% | 16% |

*Table 5: Comparison in Eb/N0 and capacity gains of 4 SCCC versus 256 CC for voice and low data rate, low quality services. A maximum of 10 and 100 iterations of the decoding algorithm.*

| BER | 1.00E-03 | | 1.00E-04 | | 1.00E-05 | |
|---|---|---|---|---|---|---|
| Block size | Gain in Eb/N0 [dB] | Gain in Capacity | Gain in Eb/N0 [dB] | Gain in Capacity | Gain in Eb/N0 [dB] | Gain in Capacity |
| **160 (100 it.)** | 0.49 | 10.67% | 0.96 | 19.83% | 1.32 | 26.22% |
| **160 (10 it.)** | 0.10 | 2.28% | 0.56 | 12.09% | 0.85 | 17.80% |

In Table 3 and Table 4 we report the numerical results obtainable from Figure 11 and Figure 9. The second (fourth and sixth) columns provides the difference in $E_b/N_0$ (in dB) between 8-PCCC and 4-SCCC, and the third (fifth and seventh) columns reports the gain in capacity induced by the gain in signal-to-noise ratio according to a formula based on very simplifying assumptions. Finally, the eighth column reports the gain of 4-SCCC in ASIC implementation complexity measured as area saving.

Table 5 reports analogous comparisons between 4-SCCC and 256-state convolutional code for voice and low data rate, low quality services. In terms of complexity, we do not report results here, because in any case we are comparing a single code with a situation in which the same code, plus the 256 CC are needed, so that the complexity of the Viterbi decoder adds to the one with the single 4-SCCC code.

In Table 5 we have reported the 4-SCCC performance using both 10 and 100 as the maximum number of iterations of the decoding algorithm. Notice that using 100 iterations as the maximum number significantly improves the performance (and the capacity), without affecting significantly the power consumption, since the average number of iterations required (less than 10) remain almost the same. On the other hand, 100 iterations, for these data rates, are widely within the reach of any reasonable decoder implementations (see next Table 6 which reports the maximum number of iterations for data rates of 4 and 8 kbit/s in a VLSI implementation, based on the architecture described in [5] that has been used to obtain the complexity measures shown in Figure 1 and using a clock of 50MHz).

*Table 6: Maximum number of iterations at 50 MHz (4SCCC)*

| **Block size / Data rate** | ***4 kbit/s*** | ***8 kbit/s*** |
|---|---|---|
| ***80*** | 2,632 | 1,316 |
| ***160*** | 3,448 | 1,724 |

From Table 3 and Table 5 (and figures) we can derive the following important system observations:

- The 4-SCCC yields uniformly good performance for all block sizes and quality of services. As compared to 8-PCCC, it is always superior for the bit error rate of $10^{-8}$, yielding a gain in capacity up to 20%. For the bit error rate of $10^{-7}$, it yields capacity gains up to 13. For the bit error rate of $10^{-6}$, it is better for the block size of 80 and looses no more than 0.25 dB (5% loss in capacity) for the larger block sizes. If the two codes are compared in terms of frame, instead of bit, error rates, then 4-SCCC shows even superior performance with respect to 8-PCCC.

- As compared with the 256-state convolutional code, it yields a gain of about 0.49 dB (10% gain in capacity) at a bit error rate of $10^{-3}$ (for a block size of 160), and even better at lower bit error rates. For a block size of 80, the two codes show the same performance at a bit error rate of $10^{-3}$, whereas significant gains, up to 10% in capacity are provided by the 4-SCCC at lower bit error rates. As a consequence, 4-SCCC can be used without requiring the presence of the convolutional code.

- The implementation complexity of 4-SCCC is lower than for 8-PCCC for all block sizes, yielding a saving close to 90 % for the lower block sizes, of about 80 % for a size of 320, 70 % for 640, and 35 % for 1,280. At the largest block size (5,120), the saving becomes to 16 %.

- The performance curves versus block size show a marginal improvement for block sizes above 1,280. In fact, increasing from 1,280 to 5,120 yields about 0.4 dB of improvement, whereas going from 2,560 to 5,120 yields an improvement less than 0.2 dB. On the other hand, the implementation complexity increases more than twice. As a consequence, it seems more effective, for a better performance/complexity trade-off, to limit the information block size to 1,280.

- For non real time services (as an example packet data) one could use either the FEC or ARQ option. When using a rather weak code, like a convolutional code alone, the ARQ option allows obtaining low bit error rates at low $E_b/N_0$, at the expense of the data throughput. This is because ARQ can base its strategy on relatively high FER ($10^{-2}$-$10^{-3}$ ) to obtain low BER ($10^{-7}$ - $10^{-8}$ ). On the other hand, when a very powerful code (like the 4-SCCC) is available, showing almost vertical curves of BER versus $E_b/N_0$ down to BER of $10^{-8}$ without error floors, it is convenient to use an FEC strategy, yielding BER of $10^{-8}$ without sacrificing neither throughput nor $E_b/N_0$. As a consequence, 4-SCCC can be effectively used also for non real time services, together with an FEC strategy.

## Sensitivity to puncturing for rate matching

We have also performed a preliminary investigation of the sensitivity of 4-SCCC and 8-PCCC to the puncturing required for the rate matching. The following results refer to a puncturing depth such as to reduce the code rate from 1/3 to ½.

We have chosen as puncturing pattern two different cases:

- A random puncturing pattern varying from block to block, corresponding to a completely uncorrelated operation of the rate matching puncturing with respect to the encoder (curves denoted by (R) in the figures).
- A periodic puncturing, one bit every three, corresponding to the presently proposed rate matching algorithm directly applied to the encoded blocks encoder (curves denoted by (P) in the figures).

Figure 13 and Figure 14 report the results for the two extreme block sizes of 80 and 5120 for the cases of 4-SCCC and 8-PCCC. The results for the unpunctured rate 1/3 code are also reported for comparison. The case of periodic puncturing for the 8-PCCC code has been chosen so as to puncture one of the two parity check bits. In fact, puncturing the systematic bit would lead to a quasi-catastrophic code.

From the curves we see that the loss for puncturing lies between 0.79 and 0.9 dB for SCCC, and between 0.85 and 5.5 for PCCC in the case of N=5120. For the case N=80 SCCC looses from 0.95 to 1.66 dB whereas PCCC looses from 2.03 to 2.09 dB. The large loss for periodic puncturing in the case of 8-PCCC stems from the lack of the turbo effect in this case.

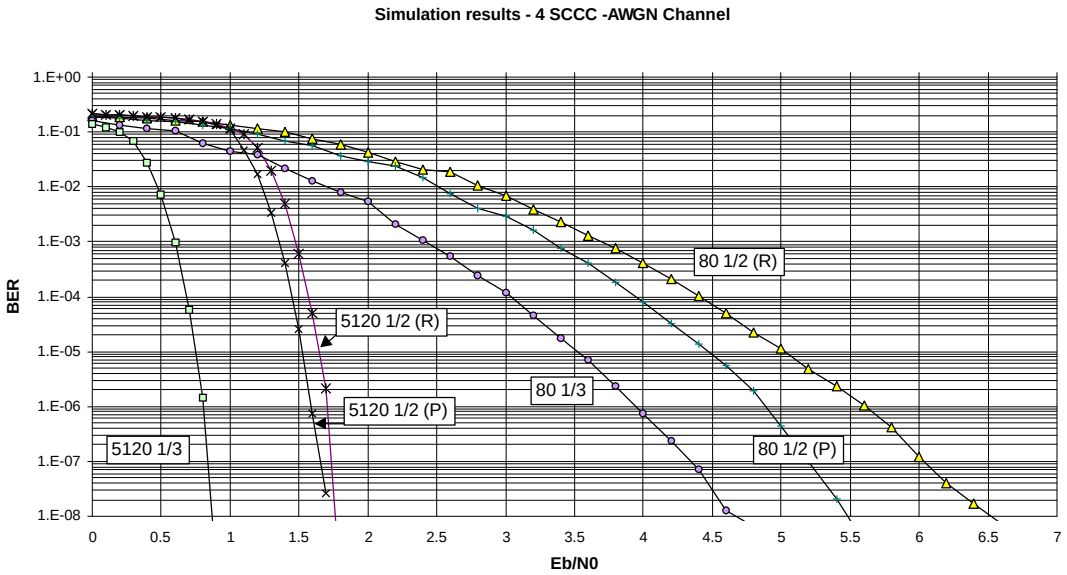


*Figure 13: Bit error probability of the 4-state SCCC for block sizes of 80 and 5120 in the presence of random and periodic puncturing leading to a rate ½ code. The curves for rate 1/3 (unpunctured) is also reported.*

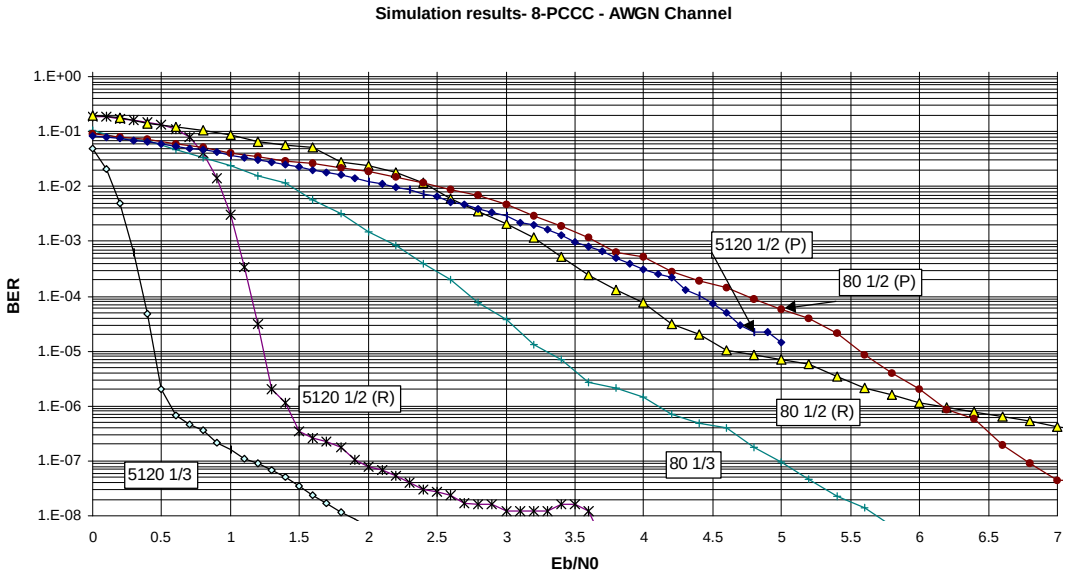


*Figure 14: Bit error probability of the 8-state PCCC for block sizes of 80 and 5120 in the presence of random and periodic puncturing leading to a rate ½ code. The curves for rate 1/3 (unpunctured) is also reported.*

## Conclusions

Based on the set of results presented in this document, we can conclude that the coding scheme based on the serial concatenation of two 4-state convolutional codes with interleaver (4-SCCC) is a very strong candidate for all data rates, block sizes and quality of services. Indeed:

1. It offers good performance for all block sizes (80 to 5,120) and all bit and frame error rates ($10^{-3}$ – $10^{-8}$), thus making unnecessary to standardize several different codes for the various data rates, block sizes and quality of services.
2. For real time data and non real time data ( packet data services) which required BER<$10^{-6}$ a capacity gain up to 20% is achieved.
3. For all block sizes, the free distance of the 4-SCCC is better than the other proposed solutions (256-state convolutional code and 8-state PCCC). This makes 4-SCCC very robust to further puncturing for rate (or service-specific) matching.
4. For voice, which typically requires a bit error rate around $10^{-3}$, 4-SCCC is almost as good as 256 CC, and for BER=$10^{-3}$ – $10^{-5}$ , block size 160 bits, 4-SCCC offers a capacity gain of 10% -26%.
5. In terms of complexity, 4-SCCC is significantly better than the 8-PCCC (from 90 % to 16 % less complex depending on the block size), and does not require extra codes for low data rates, or high-quality services, like a solution based on 8-PCCC would indeed require.
6. The only price to be paid for choosing this universal code would be a penalty of no more than 0.25 dB for the large block sizes at a bit error rate of $10^{-6}$ for rate 1/3 codes.
7. A preliminary investigation of the sensitivity to puncturing for rate matching shows the robustness of the 4-SCCC, which does not seem to present significant losses both for random and periodic, or regular, deterministic puncturing.

## References

[1] 3GPP (S1.12) v1.0.1
[2] UMTS 22.05 v3.3.0
[3] HNS TSGR1#2(99)101.
[4] Lucent TSGR1#2(99)110.
[5] Lucent TSGR1#2(99)038.