Correct Undetected Errors with List Decoding in ARQ Error-control Systems

Jingzhao Wang  
Department of Computer Science and Engineering  
Shanghai Jiao Tong University  
Shanghai 200240, China  
Email: wangzhe.90@sjtu.edu.cn

Yuan Luo  
Department of Computer Science and Engineering  
Shanghai Jiao Tong University  
Shanghai 200240, China  
Email: yuanluo@sjtu.edu.cn

Abstract—Undetected errors are important for linear codes, which are the only type of errors after hard decision and automatic-repeat-request (ARQ), but do not receive much attention on their correction. In concatenated channel coding, sub-optimal source coding and joint source-channel coding, constraints among successive codewords may be utilized to improve decoding performance. In this paper, list decoding is used to correct the undetected errors. The benefit proportion of the correction is obviously improved especially on Hamming codes and Reed-Muller codes, which achieves about 40% in some cases. But this improvement is significant only after the selection of final codewords from the lists based on the constraints among the successive transmitted codewords. The selection algorithm is investigated here to complete the list decoding program in the application of Markov context model. The performance of the algorithm is analysed and a lower bound of the correctly selected probability is derived to determine the proper context length.

I. INTRODUCTION

Error detection using block codes combined with repeat requests is a widely used method of error control in communication systems[6], which is called automatic-repeat-request (ARQ) scheme.

Undetected errors are the only type of errors after hard decision and automatic-repeat-request (ARQ), where the sent codeword is transmitted into another codeword. List decoding was introduced independently by Elias [1] and Wozencraft [11] in the late 1950’s. For a received vector, the decoder doesn’t output a single codeword, but outputs a list of possible codewords. It is considered to be list decoded correctly if the sent codeword is in the list. This paper uses list decoding to correct undetected errors of ARQ systems and give a list of possible codewords for each received codeword after retransmissions.

In some cases, there may be constraints among successive codewords, i.e. the input codewords are dependent. For example, when concatenated codes [2] are used in channel coding, the codewords of the inner code must be constrained by the outer code. In joint source-channel coding, there also be memory structure inherent with in the sequence output from the source [12]. Moreover, in the model for storage or communication, many files are not compressed or compressed sub-optimally in source coding [4,5,10]. In this situation, there may also be constraints among the successive transmitted codewords, where redundancy mining can be used for error correction, which is considered in [4,5]. Inspired by these works, we assume the code context model is a Markov chain which is an elementary stochastic model used in natural language processing models, and investigate an algorithm to select codewords from the lists.

This paper is organized as follows. Section II presents some necessary preliminaries and notations. In Section III the error correcting probability by list decoding of linear codes is analysed. And Hamming codes and Reed-Muller codes are investigated as numerical examples where the benefit proportion of list decoding can achieve about 40% in some cases. In Section IV an algorithm based on context is investigated to complete list decoding program, with which the most probable codewords are selected from the lists. The algorithm performance is analysed in Section V. Finally, we conclude this paper in Section VI.

II. PRELIMINARIES AND NOTATIONS

The communication channel used in this paper is binary symmetric channel (BSC) with crossover probability p. The encoder is a one-to-one map function from the message set \( M = \{0,1,...,M-1\} \) to the codebook \( C \), where \( C \) is a subset of \( \{0,1\}^n \) and \( n \) is known as the blocklength of code \( C \).

The Hamming weight of a vector \( x \) and \( d_H(x,y) \) is the Hamming distance between two vectors \( x, y \in \{0,1\}^n \).

For a binary \( [n, k, d] \) linear code \( C \) with minimum distance \( d = \min\{w_H(x)|x \in C, x \neq 0\} \), the weight distribution is defined as

\[
A_i^w = A_i^{w(C)} = \#\{x \in C|w_H(x) = i\}
\]

and the weight distribution function is defined as

\[
A_{w}^{w}(x, y) = \sum_{i=0}^{n} A_i^w x^{n-i} y^i.
\]

III. THE ERROR PROBABILITY OF ARQ SYSTEMS WITH LIST DECODING

In this section, we first analyse the error correcting probability by list decoding of linear codes, which can be calculated
by Proposition 1. Then the error probability of ARQ systems is calculated in Proposition 2. In Subsection III-A and III-B, the benefit proportion of error probability after list decoding is illustrated in Fig. 1 and Fig. 2, where Hamming codes and Reed-Muller codes are investigated as numerical examples.

Assume $X = (X_1, X_2, ..., X_n) \in \{0, 1\}^n$ is sent and $Y = (Y_1, Y_2, ..., Y_n) \in \{0, 1\}^n$ is received, where $X$ and $Y$ are vectors of random variables. Let $B(y, r) = \{ z \in \{0, 1\}^n : y \in \{0, 1\}^n, d_H(z, y) \leq r \}$ denote the sphere of radius $r$ about $y$ and consider the conditional probability as follows:

$$\Pr\{Y \in C \text{ and } Y \neq 0^n, 0^n \in B(Y, r) | X = 0^n\} = \Pr\{Y \in C \text{ and } Y \neq 0^n, 0^n \in B(0^n, r) | X = 0^n\}, \quad (1)$$

where $r \geq d$. $P_{\text{List}}(r)$ is called the error correcting probability by list decoding, which is also the reduced probability of undetected error after list decoding, i.e.,

$$P_{\text{List}}(r) = \sum_{c \in C, c \neq 0^n} \Pr\{Y = c, c \in B(0^n, r) | X = 0^n\}, \quad (2)$$

where $r \geq d$. $P_{\text{List}}(r)$ is the list size of decoding radius $r$ is large and be at most a polynomial in the blocklength. So this case, it is defined to be list decoded correctly. For all the codewords in $C$, the probability in (1) can be described as follows:

After list decoding in ARQ system, the $P_{\text{ue}}$ is reduced to $P_{\text{List}} = P_{\text{ue}} - P_{\text{List}}^c$ and the error probability changes from (3) to

$$P_{\text{e}}^{\text{ARQ, List}} = \frac{P_{\text{List}}}{P_{\text{ue}} + P_{c}}. \quad (4)$$

The benefit proportion is defined by

$$P_b = \frac{P_{\text{e}}^{\text{ARQ}} - P_{\text{e}}^{\text{ARQ, List}}}{P_{\text{e}}^{\text{ARQ}}}. \quad (5)$$

The results (3), (4) and (5) form the following proposition.

**Proposition 2.** If the crossover probability of BSC is $p$, it is clear that $P_{e} = (1-p)^n$, $P_{\text{ue}} = A_w^n (1-p, p) - P_{e} = A_w^n (1-p, p) - (1-p)^n$. The probability of undetected error after list decoding is $P_{\text{List}} = P_{\text{ue}} - P_{\text{List}}$. After ARQ retransmissions, assume all the received vectors are codewords, then the error probability is

$$P_{\text{e}}^{\text{ARQ}} = \frac{P_{\text{ue}}}{P_{\text{ue}} + P_{c}}$$

$$= \frac{A_w^n (1-p, p) - (1-p)^n}{A_w^n (1-p, p)},$$

the error probability after list decoding is

$$P_{\text{e}}^{\text{ARQ, List}} = \frac{P_{\text{List}}}{P_{\text{ue}} + P_{c}}$$

$$= \frac{A_w^n (1-p, p) - (1-p)^n - A_w^n p^d \cdot (1-p)^{n-d}}{A_w^n (1-p, p)}$$

and the benefit proportion is

$$P_b = \frac{P_{\text{e}}^{\text{ARQ}} - P_{\text{e}}^{\text{ARQ, List}}}{P_{\text{e}}^{\text{ARQ}}}$$

$$= \frac{A_w^n p^d \cdot (1-p)^{n-d}}{A_w^n (1-p, p) - (1-p)^n}.$$
in the following part, we use the state in ARQ systems, all the received vectors are codewords. So mapped to a state of the process. After retransmissions in states, see the following part.

codewords from lists is equivalent to the determination of the selected from the lists based on the context. The selection of the corresponding codeword and give a list of all possible on

with block length $M$. The appearance of a codeword may be affected by former or latter codewords, which is called the context of a code. For instance, the appearance of each English letter in a word depends on its former and latter letters. At the same time, context structure. The process of CLDA is shown in Algorithm 1 and Example 1, with which the most probable codewords are selected from the lists.

The context of a code can be represented by a stochastic process as a Markov chain and let $P_{ij}$ denote the one-step transition probability from state $i$ to state $j$, then
\[
P_{ij} = \Pr\{X_{t+1} = j | X_t = i\},
\]
for all states $i, j \in \mathcal{M}$ and all $t \geq 0$. Let $\mathbf{P}$ denote the transition matrix of $P_{ij}$, then
\[
\mathbf{P} = (P_{ij})_{M \times M}(0 \leq i, j \leq M - 1).
\]

The context of a code can be represented by $\mathbf{P}$, with which Algorithm 1 (CLDA) will select the most probable queue of codewords. To describe CLDA, some definitions are given as follows.

**Definition 1.** A queue of successive codewords with context constraint is called a codeword sentence. The appearance probability of a codeword sentence is dubbed codeword sentence weight.

Then, the weight of $N$-codeword sentence $(i_0, i_1, \ldots, i_{N-1})$ is
\[
\Pr\{X_0 = i_0, \ldots, X_{N-1} = i_{N-1}\} = \Pr\{i_0\} \cdot \Pr\{i_1 | i_0\} \cdot \cdots \cdot \Pr\{i_{N-2} | i_{N-3}\} | i_0\}
\]
\[
= \pi_{i_0} \cdot P_{i_1, i_0} \cdot \cdots \cdot P_{i_{N-2}, i_{N-3}},
\]
where let $Pr_{i_0} \triangleq \Pr\{X_0 = i_0\} \triangleq \pi_{i_0}$ denote the priori probability of $i_0$ and let $Pr_{i_{k+1} | i_k} \triangleq \Pr\{X_{k+1} = i_{k+1} | X_k = i_k\}$.

For linear codes, every given list contains the same number of codewords and $L$ is the list size. Let $\mathbf{r} = (r_0, r_1, \ldots, r_{N-1})$ denote the received codeword sentence and $N$ is the codeword sentence length. Let $\mathbf{l} = (l_0, l_1, \ldots, l_{N-1})$ denote the codeword sentence selected from the successive lists, then the problem of finding the most probable $N$-codeword sentence in the lists is equivalent to compute
\[
\arg\max_{(l_0, l_1, \ldots, l_{N-1}) \in L_0 \times L_1 \times \cdots \times L_{N-1}} \Pr\{(l_0, l_1, \ldots, l_{N-1})\},
\]
where $L_0, L_1, \ldots, L_{N-1}$ are successive lists given by $\mathbf{r}$ as follows:
\[
L_i = \mathcal{L}(r_i), 0 \leq i \leq N - 1,
\]
\[
\mathcal{L}(r_i) = \{c \in C : d_H(c, r_i) \leq d\}.
\]

Let function $T(l_i)$ compute the maximal codeword sentence weight when some codeword $l_i$ is selected form List $L_i$ and appended to the previously selected codeword sentence $(l_0, l_1, \ldots, l_{i-1})$. For $i \in [2, N - 1]$, we have
\[
T(l_i) \triangleq \max_{(l_0, l_1, \ldots, l_{i-1}) \in L_0 \times L_1 \times \cdots \times L_{i-1}} \Pr\{(l_0, l_1, \ldots, l_{i-1})\} \Pr\{l_i\}.
\]
Then
\[
T(l_i) = \max_{(l_0, l_1, \ldots, l_{i-1}) \in L_0 \times L_1 \times \cdots \times L_{i-1}} \Pr\{(l_0, l_1, \ldots, l_{i-1})\} \Pr\{l_i\}.
\]
\[
= \max_{i_{i-1} \in L_{i-1}} \left(\Pr\{l_i | l_{i-1}\} \cdot \max_{(l_0, l_1, \ldots, l_{i-2}) \in L_0 \times L_1 \times \cdots \times L_{i-2}} \Pr\{(l_0, l_1, \ldots, l_{i-2})\}\right)
\]
\[
= \max_{i_{i-1} \in L_{i-1}} \Pr\{l_i | l_{i-1}\} T(l_{i-1})
\]
and $T(l_0) = \Pr\{l_0\} = \pi_{l_0}$.

The derived recurrence suggests that the optimization problem can be solved with dynamic programming algorithm specified in Algorithm 1 (CLDA), which is adapted from the Viterbi decoding. The final solution is computed iteratively, starting from $T(t_1)$ according to the recurrence. When the last iteration is finished, we trace back along the path with the maximal codeword sentence weight, selecting the most probable codeword sentence.

Let $L = (l_{ij})_{L \times N}(0 \leq i \leq L - 1, 0 \leq j \leq N - 1)$ store the values of each node. Let $P = (P_{ij})_{M \times M}(0 \leq i, j \leq M - 1)$ denote the codeword transition probability matrix and $\Pi = (\pi_0, \pi_1, ..., \pi_{M-1})^T$ denote the priori probability of each codeword, which are decided by the context of the codewords. The CLDA is as follows:

**Algorithm 1 (CLDA).**

In the following four steps, first, $P$, $L$ and $\Pi$ defined above can be used to calculate a weight matrix $T$ and a jumping point matrix $D$. Then, matrix $R$, which stores the jumping points of the path with length $N$ and the largest weight, is determined by $T$ and $D$. At last, the final codewords after context list decoding are stored in $F$ calculated by $R$ and $L$.

1. Initialize matrix $T = (t_{ij})_{L \times N}$ where $t_{ij}$ stores the maximum codeword sentence weight from the nodes in List 0 to the $i$th node in the current List $j$. The first column of $T$ is initialized with $(\pi_{l_0,0}, \pi_{l_1,0}, ..., \pi_{l(L-1),0})^T$, namely, $t_{i,0} = \pi_{l_i,0}(0 \leq i \leq L - 1)$ and others with zeroes.

2. From List 0 to List $N - 1$, compute the codeword sentence weight:

$$t_{ij} = \max_{0 \leq k \leq L-1} \{t_{k,j-1} \cdot P_{k,j-1,ij}\}, \quad (9)$$

where $0 \leq i \leq L - 1, 1 \leq j \leq N - 1$.

The corresponding $k$ which achieves the maximum value is stored in matrix $D = (x_{ij})_{L \times N}$:

$$x_{ij} = \arg \max_{0 \leq k \leq L-1} \{t_{k,j-1} \cdot P_{k,j-1,ij}\}, \quad (10)$$

where $0 \leq i \leq L - 1, 1 \leq j \leq N - 1$.

**Remark.**

- $k$ is the row (node) index and $j$ is the column (list) index;
- $l_{ij}$ is the value (codeword) of the $i$th node in List $j$;
- $x_{ij}$ is the jumping point from List $j - 1$ to the $i$th node in List $j$, which reaches the largest weight to current node.
- We select one codeword randomly if there are more than one achieving the maximum value.

3. Calculate $R = (r_0, r_1, ..., r_{N-1})$, where $r_{N-1} = \arg \max_{0 \leq k \leq L-1} t_{i,N-1}$ and $r_{k-1} = x_{r_k,k}$ ($k$ is from $N - 1$ to $0$).

4. Calculate $F = (f_0, f_1, ..., f_{N-1})$, where $f_k = l_{r_k,k}$ ($k$ is from $0$ to $N - 1$). Return $F$.

The complexity of CLDA is $O(L^2 N)$, where $L$ is the list size and $N$ is the length of codeword sentence. An example of CLDA is illustrated as follows.

---

**Fig. 2. The Numerical Results of Reed-Muller Codes**
Example 1. Let the codeword transition probability matrix $P$ be
\[
P = \begin{pmatrix}
  1/3 & 2/3 & 0 \\
  2/9 & 5/9 & 2/9 \\
  0 & 2/3 & 1/3
\end{pmatrix}
\]

Note that, $P$ is from the context and has no relation with BSC. Meanwhile, let the priori probability of each codeword be $\Pi = (1/5,3/5,1/5)^T$. There are 3 successive lists with list size 3, which is stored in $L$.

\[
L = \begin{pmatrix}
  0 & 0 & 0 \\
  1 & 1 & 1 \\
  2 & 2 & 2
\end{pmatrix}
\]

1. Initialize the first column of $T$ with $\Pi = (1/5,3/5,1/5)^T$.
2. For each 2-codeword sentence from List 0 to List 1, and $\{\}^{[3]}$ are calculated as follows: $\pi_0*P_{00} = 1/15$, $\pi_1*P_{10} = 2/15$, $\pi_2*P_{20} = 0$. Then, record the biggest value $2/15$ in $T$ and corresponding jumping point $x_{01}; t_{01} = 2/15$, $x_{01} = 1$.

Similarly, calculate $\pi_0*P_{01} = 2/15$, $\pi_1*P_{11} = 1/3$, $\pi_2*P_{21} = 2/15$. Then $t_{11} = 1/3$, $x_{11} = 1$.

And calculate $\pi_0*P_{02} = 0$, $\pi_1*P_{12} = 2/15$, $\pi_2*P_{22} = 1/15$. Then $t_{21} = 2/15$, $x_{21} = 1$.

For each 3-codeword sentence from List 0 to List 3, compute $t_{01}*P_{00} = 2/45$, $t_{11}*P_{10} = 2/27$, $t_{21}*P_{20} = 0$, and record the biggest value $2/27$ and corresponding index 1. Then $t_{02} = 2/27$, $x_{02} = 1$.

Similarly, $t_{12} = 5/27$, $x_{12} = 1$, $t_{22} = 2/27$, $x_{22} = 1$.

3. We can find the most probable sentence using $T$ and $A$, as shown in Algorithm 1. $t_{12}$ is the maximum among $t_{02}$, $t_{12}$ and $t_{22}$. Then $r_2 = 1$ and $r_1 = x_{12} = 1$, $r_0 = x_{11} = 1$. Using $R = (1,1,1)$, we get $F = (t_{10}, t_{11}, t_{12}) = (1,1,1)$ which is the final codeword sentence after context list decoding.

V. PERFORMANCE OF CLDA

As the ARQ system transmits messages through BSC with crossover probability $p$, the probability of retransmission each time is
\[
Pr_{\text{trans}} = 1 - A_C^w(1 - p, p),
\]
where we assume the sent codeword is $(0,0,...,0)_n$. In the following paragraphs, $\alpha(\alpha \geq 0)$ denotes the number of retrasmisions for receiving a legal codeword.

Let $s = (s_0, s_1, ..., s_{N-1})$ be the sent codeword sentence. To guarantee that $s$ can be selected from the $N$ lists in the decoding, it is obvious that $Pr(s)$ should be the biggest among all the sentences in $L_0 \times L_1 \times ... L_{N-1}$. This constraint depends not only on the codeword sentence but also on the transition probability matrix $P$.

Assumption. Assume that $Pr(s)$ is the biggest among all the codeword sentence in $L(s_0) \times L(s_1) \times ... L(s_{N-1})$. Then if received sentence $r$ equals to $s$, the CLDA selects $s$ obviously. The following paragraphs try to consider the situation that $r \neq s$.

Assume $s_i \neq r_i$ for some $0 \leq i \leq N - 1$. If the list given by $s_i$ is the same with the list given by $r_i$, $s$ will be selected from the lists correctly, otherwise $s$ may be selected wrongly.

Let $S(s_i)$ denote the set of codewords which give the same list with $s_i$, namely
\[
S(s_i) = \{ c \in C : L(c) = L(s_i) \}. \tag{11}
\]

We use $E_c$ to denote the event that $s$ is selected from the lists correctly. Meanwhile, let $E_c(i)(0 \leq i \leq N - 1)$ denote the event that $s_i$ is selected correctly from $L_i$. Then a lower bound of $Pr\{E_c\}$ is as follows:
\[
Pr\{E_c\} = Pr\{ \bigcap_{i=0}^{N-1} E_c(i) \} = \prod_{i=0}^{N-1} Pr\{E_c(i)\}
\]
\[
\geq \prod_{i=0}^{N-1} \left( 1 - \sum_{\alpha=0}^{\infty} P_{\text{trans}}^\alpha Pr\{r_i \in S(s_i)\} \right) = (1 - P_{\text{trans}})N \prod_{i=0}^{N-1} (P_c + (|S(s_i)| - 1)p^d(1 - p)^{N-d}). \tag{12}
\]

Note that $Pr\{r_i \in S(s_i)\}$ is calculated based on the definition of $L(\cdot)$ and $S(\cdot)$, see (8) and (11).

Numerical results for (12) are shown in Fig. 3 where the lower bound of $P_{E_c}$ is evaluated with RM(3,5) code. In Fig. 3 (a), the lower bound of $P_{E_c}$ decreases when the crossover probability $p$ of BSC varies from 0 to 0.1 for codeword sentence length $N = 10,20$. It can be seen that the bound is better for small $p$, i.e. $p \in (0,0.06)$, which is partly shown in Fig. 3 (b). The lower bound is about 0.98 for $p \in (0.04,0.06)$ which may be practical. Fig. 3 (c) shows that the lower bound of $P_{E_c}$ decreases when the codeword sentence $N$ varying from 5 to 30 and $p = 0.06$. For small $N$, $P_{E_c}$ may be relatively large, however, the context may not work well in CLDA. When $p$ and $P_{E_c}$ is fixed, the largest $N$ can be calculated out with (12), which is meaningful for the selection of the proper codeword sentence length.

When $p$ is fixed, the lower bound can be used to calculated the proper $N$ needed to achieve required correctly selected probability $P_{E_c}$.

In the following part, we try to analyse the average correctly selected probability denoted by $P_{\text{average}}$.

The probability of all the $M^N$ codeword sentences can be calculated out with $P$. It is convenient to order all the codeword sentences by the probability from large to small.

Let $s_i = (s_{i0}, s_{i1}, ..., s_{i,N-1})(1 \leq i \leq M^N)$ denote the ordered codeword sentences, where $Pr\{s_i\}$ is decreasing by $i$ from 1 to $M^N$, namely, $Pr\{s_1\}$ is largest and $Pr\{s_{M^N}\}$ is the smallest.

For each sent codeword sentence $s_i$, the probability that $s_i$ is selected correctly varies with $Pr\{s_i\}$. Consider $s_i$ and $Pr\{s_i\}$ for different $1 \leq i \leq M^N$ respectively as follows:

1. $s_1$ and $Pr\{s_1\}$:
To be convenient, we assume \( \bigcup_{k=1}^{N} \mathcal{L}(s_k) = \emptyset \).

This allows us to calculate the average correctly selected probability as follows:

\[
P_{\text{average}} = \frac{\sum_{i=1}^{M} (\Pr\{s_i \text{ is sent}\} \cdot \Pr\{s_i \text{ is selected correctly} | s_i \text{ is sent}\})}{(1 - P_{\text{retrans}})^N}.
\]

\[
P_{\text{average}} = \frac{1}{(1 - P_{\text{retrans}})^N} \sum_{i=1}^{M} \prod_{k=0}^{N-1} P_{retrans}(r_k, s_{i,k}) (1 - p)^{n - d_H(r_k, s_{i,k})}.
\]

To be convenient, we assume \( \bigcup_{k=1}^{N} \mathcal{L}(s_k) = \emptyset \).

This allows us to calculate the average correctly selected probability as follows:

\[
P_{\text{average}} = \frac{1}{(1 - P_{\text{retrans}})^N} \sum_{i=1}^{M} \prod_{k=0}^{N-1} P_{retrans}(r_k, s_{i,k}) (1 - p)^{n - d_H(r_k, s_{i,k})}.
\]

To be convenient, we assume \( \bigcup_{k=1}^{N} \mathcal{L}(s_k) = \emptyset \).

This allows us to calculate the average correctly selected probability as follows:

\[
P_{\text{average}} = \frac{1}{(1 - P_{\text{retrans}})^N} \sum_{i=1}^{M} \prod_{k=0}^{N-1} P_{retrans}(r_k, s_{i,k}) (1 - p)^{n - d_H(r_k, s_{i,k})}.
\]

To be convenient, we assume \( \bigcup_{k=1}^{N} \mathcal{L}(s_k) = \emptyset \).

This allows us to calculate the average correctly selected probability as follows:

\[
P_{\text{average}} = \frac{1}{(1 - P_{\text{retrans}})^N} \sum_{i=1}^{M} \prod_{k=0}^{N-1} P_{retrans}(r_k, s_{i,k}) (1 - p)^{n - d_H(r_k, s_{i,k})}.
\]

To be convenient, we assume \( \bigcup_{k=1}^{N} \mathcal{L}(s_k) = \emptyset \).

This allows us to calculate the average correctly selected probability as follows:

\[
P_{\text{average}} = \frac{1}{(1 - P_{\text{retrans}})^N} \sum_{i=1}^{M} \prod_{k=0}^{N-1} P_{retrans}(r_k, s_{i,k}) (1 - p)^{n - d_H(r_k, s_{i,k})}.
\]

To be convenient, we assume \( \bigcup_{k=1}^{N} \mathcal{L}(s_k) = \emptyset \).

This allows us to calculate the average correctly selected probability as follows:

\[
P_{\text{average}} = \frac{1}{(1 - P_{\text{retrans}})^N} \sum_{i=1}^{M} \prod_{k=0}^{N-1} P_{retrans}(r_k, s_{i,k}) (1 - p)^{n - d_H(r_k, s_{i,k})}.
\]

To be convenient, we assume \( \bigcup_{k=1}^{N} \mathcal{L}(s_k) = \emptyset \).

This allows us to calculate the average correctly selected probability as follows:

\[
P_{\text{average}} = \frac{1}{(1 - P_{\text{retrans}})^N} \sum_{i=1}^{M} \prod_{k=0}^{N-1} P_{retrans}(r_k, s_{i,k}) (1 - p)^{n - d_H(r_k, s_{i,k})}.
\]

To be convenient, we assume \( \bigcup_{k=1}^{N} \mathcal{L}(s_k) = \emptyset \).

This allows us to calculate the average correctly selected probability as follows:

\[
P_{\text{average}} = \frac{1}{(1 - P_{\text{retrans}})^N} \sum_{i=1}^{M} \prod_{k=0}^{N-1} P_{retrans}(r_k, s_{i,k}) (1 - p)^{n - d_H(r_k, s_{i,k})}.
\]

To be convenient, we assume \( \bigcup_{k=1}^{N} \mathcal{L}(s_k) = \emptyset \).

This allows us to calculate the average correctly selected probability as follows:

\[
P_{\text{average}} = \frac{1}{(1 - P_{\text{retrans}})^N} \sum_{i=1}^{M} \prod_{k=0}^{N-1} P_{retrans}(r_k, s_{i,k}) (1 - p)^{n - d_H(r_k, s_{i,k})}.
\]

To be convenient, we assume \( \bigcup_{k=1}^{N} \mathcal{L}(s_k) = \emptyset \).

This allows us to calculate the average correctly selected probability as follows:

\[
P_{\text{average}} = \frac{1}{(1 - P_{\text{retrans}})^N} \sum_{i=1}^{M} \prod_{k=0}^{N-1} P_{retrans}(r_k, s_{i,k}) (1 - p)^{n - d_H(r_k, s_{i,k})}.
\]

To be convenient, we assume \( \bigcup_{k=1}^{N} \mathcal{L}(s_k) = \emptyset \).

This allows us to calculate the average correctly selected probability as follows:

\[
P_{\text{average}} = \frac{1}{(1 - P_{\text{retrans}})^N} \sum_{i=1}^{M} \prod_{k=0}^{N-1} P_{retrans}(r_k, s_{i,k}) (1 - p)^{n - d_H(r_k, s_{i,k})}.
\]

To be convenient, we assume \( \bigcup_{k=1}^{N} \mathcal{L}(s_k) = \emptyset \).

This allows us to calculate the average correctly selected probability as follows:

\[
P_{\text{average}} = \frac{1}{(1 - P_{\text{retrans}})^N} \sum_{i=1}^{M} \prod_{k=0}^{N-1} P_{retrans}(r_k, s_{i,k}) (1 - p)^{n - d_H(r_k, s_{i,k})}.
\]