Error Control Mechanism Based on Reed-Solomon Code for Wireless Networks

Xinyu Wang
Tianjin University of Technology
Kai Shi (shikai0229@tjut.edu.cn)
Tianjin University of Technology
Jinsong Wang
Tianjin University of Technology
Sheng Lin
Tianjin University of Technology
Guangping Xu
Tianjin University of Technology
Fuqiang Qiao
Tianjin Sino-German University of Applied Sciences

Research

Keywords: Error Control, Reed-Solomon Codes, Burst Errors, Wireless Networks

Posted Date: July 27th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-253651/v1

Title Page

Error Control Mechanism Based on Reed-Solomon Code for Wireless Networks

Kai Shi(**Corresponding author)**
Tianjin University of Technology, 391 Binshui West Road, Xiqing District, Tianjin, China.
Email: shikai0229@tjut.edu.cn

Xinyu Wang
Tianjin University of Technology, 391 Binshui West Road, Xiqing District, Tianjin, China.
Tel.: +86-15931530628
Email: wwwlll0088@126.com

Jinsong Wang
Tianjin University of Technology, 391 Binshui West Road, Xiqing District, Tianjin, China.
Email: jswang@tjut.edu.cn

Sheng Lin
Tianjin University of Technology, 391 Binshui West Road, Xiqing District, Tianjin, China.
Email: linsheng@tjut.edu.cn

Guangping Xu
Tianjin University of Technology, 391 Binshui West Road, Xiqing District, Tianjin, China.
Email: gpxu@tjut.edu.cn

Fuqiang Qiao
Tianjin Sino-German University of Applied Sciences, College of Software and Communications, Tianjin, China.
Email: qiaofq@126.com

ABSTRACT

The reliability of information transmission has a significant influence on network performance, so it has attracted extensive attention from researchers. Many error control mechanisms have been designed and proposed in order to improve the reliability of transmission. However, during transmission in wireless networks, high bit error rate and burst errors often occur, which poses great challenges in the design of error control mechanisms. The existing mechanisms suffer from a problem of either poor error correction ability or waste of network resources. The primary aim of this study is to develop an error control mechanism based on Reed-Solomon (RS) codes, which encodes packets using RS codes, and a re-encoding algorithm is designed for reducing the coded packet length. The proposed error control mechanism can not only reduce the number of redundant bits in the transmission process but also improve the error correction ability as much as possible when burst errors occur. Therefore, both the error correction ability and the network utility are considered in this work. The proposed mechanism was verified through theoretic analysis and by experiments using the NS2 simulator. The experimental results verified the error control ability and throughput performance of the proposed mechanism.

Keywords: Error Control, Reed-Solomon Codes, Burst Errors, Wireless Networks

1. INTRODUCTION

1.1 Background and motivation

Wireless networks have been widely used in people's everyday life, allowing easy, efficient, and reliable data transmission. However, these networks have certain disadvantages, such as high bit error rate and burst errors. Thus, how to overcome these disadvantages has become an important topic in the data transmission field.

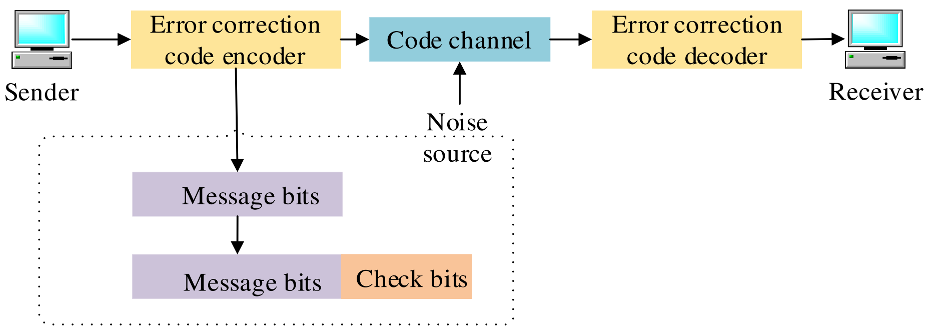


Fig. 1. Error control coding model

In recent studies, error control coding has been preferred as a solution to overcome the influence of a high bit error rate in order to improve the transmission reliability of wireless networks. Error control coding represents an error control technique that mainly serves to ensure accurate transfer of data packets by appending the redundant bits during the transmission process. The error control coding model is presented in Figure 1. Admittedly, according to [1,2], in wireless networks, the distribution of congestion events as well as errors is scattered, which affects network performance to a great extent making it vary with the bit error rate. When wireless networks are in good transmission channel conditions, the bit error rate is low, and introducing many redundant bits can result in a large waste of network resources. In contrast, when wireless networks are in bad transmission channel conditions, especially when burst errors occur, the bit error rate is very high, and it is very likely that errors will not be found and corrected in spite of using the redundant bits. In other words, the number of check bits often does not match the bit error rate.

It should be noted that the existing works have great limitations in matching the redundancy length and bit error rate, which means that a great amount of unnecessary network resources can waste during transmission. Besides, the existing mechanisms cannot improve the error correction ability under burst channels.

Therefore, two main motivations of the paper are as follows:

(1) improve the utilization rate of network resources, and

(2) improve the error correction ability when the channel is in a burst status.

In this work, improved re-encoding Reed Solomon codes are used to find out a feasible solution in order to solve the above-mentioned limitations.

1.2 Our contribution

In this paper, an improved error control mechanism is proposed to improve the error correction ability in burst channel and reduce check redundancy. The main principle of the proposed mechanism is to re-encode the check bits of the Reed-Solomon (RS) code by XOR operation and cross-check the received packets, which means that the check bits of a data packet are used to check not only the current data packet but also the two adjacent data packets.

The main contributions of this paper follow.

- An improved RS re-encoding algorithm is developed which considers the network packet loss characters. Using the proposed mechanism, the coding redundancy is reduced, and the error correction ability is improved when burst errors occur; thus the overall network performance is improved.

- Based on the proposed RS re-encoding algorithm, the packet retransmitted mechanism is designed, thus the successful transmission of data packets is guaranteed.
- The validity of the algorithm is proved through both theory and experiment.

Consequently, the proposed mechanism provides a modified and practical solution for improving the error correction ability and saving network resources. It can be used in the dynamically changing wireless network channels to ensure reliable transmission, especially in channels with frequent burst errors.

1.3 Organization of the paper

The remainder of this paper is organized as follows. Section II presents the recent related work. Section III introduces the proposed error control mechanism. Section IV evaluates the performance of the proposed mechanism by comparing it with the currently proposed mechanism. Finally, Section V presents conclusions.

2. RELATED WORK

In recent years, much effort has been directed toward improving the accuracy of data transmission in wireless networks using different schemes. According to error control methods and transmission mechanisms, the existing error control mechanisms can be divided into three categories: automatic repeat request (ARQ) error control mechanisms, forward error correction (FEC) error control mechanisms, and ARQ&FEC hybrid error control mechanisms.

ARQ

The ARQ control-based algorithms have the ability of error checking. Namely, a receiver recovers from errors by requesting the sender to resend incorrect data packets until all the packets are received correctly. The ARQ flowchart is presented in Figure 2.

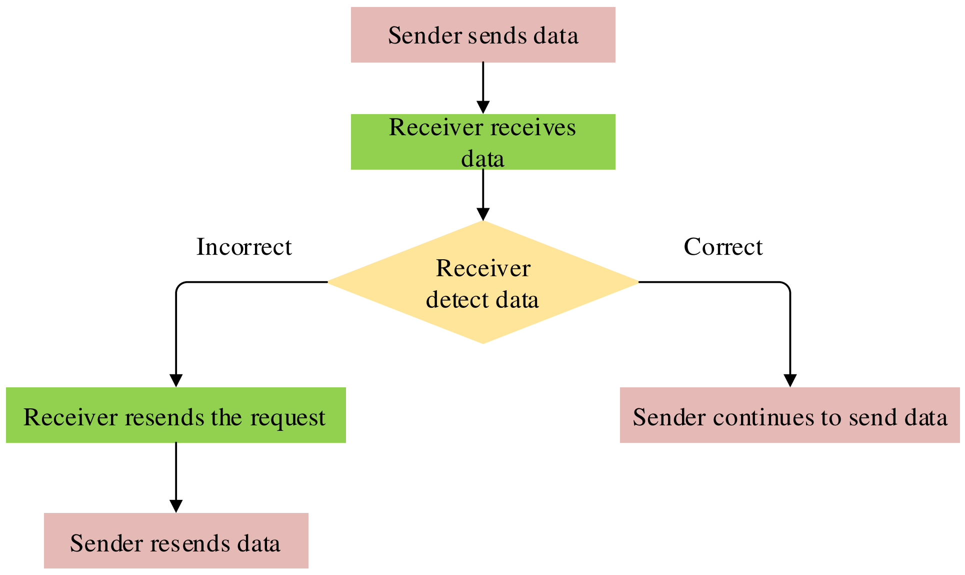


Fig. 2. The ARQ error control flowchart

In [3], a cumulative feedback-based ARQ (CF ARQ) protocol was introduced. This protocol performs well in dealing with burst errors. Namely, it has significantly less average delay under bursty feedback and a throughput gain of up to approximately 20%. In [4], the authors proposed a unified architecture that is capable of correcting burst errors, as well as random errors and erasures. The experiment results showed that this architecture could significantly improve the error correction ability compared to the traditional decoding design. In [5], three novel burst error-correcting algorithms for Reed-Solomon codes were developed. The simulation results verified that the miscorrection probability of the three algorithms decreased exponentially, but there was still the problem of network resource waste in [4,5].

Farkas et al. proposed an ARQ mechanism inspired by source coding and error correction coding, which compresses and rearranges the source code utilizing the correlation with the error correction code [6]. The main advantage of their mechanism is that it can reduce the transmission length. One example based on the exploitation of the Hamming code showed that this ARQ mechanism had a higher throughput rate than traditional ARQ methods. However, its major shortcoming is that the data packet decoding process is time-consuming, so it is not the best choice for packets with a large number of errors. Furthermore, an improved ARQ mechanism was presented in [7]. Their mechanism can obtain accurate time and position of errors, which makes it different from the previously reported solutions, and can be used for timely and accurate message transmission. However, their scheme does not perform well in reducing the length of data packets; also, it does not consider message transmission.

FEC

The FEC mechanism automatically corrects errors that occurred in the process of data transmission by adding more redundant codes, thus avoiding the retransmission process. After redundant coding, the initial message bits are appended with check bits. There are five possible scenarios in the transmission process: (1) there is no error, (2) errors occurred but can be successfully corrected, (3) errors occurred and cannot be successfully corrected, (4) part of the message is lost but can be successfully corrected, (5) part of the message is lost and cannot be successfully corrected. The flowchart of the FEC error control mechanism is shown in Figure 3.

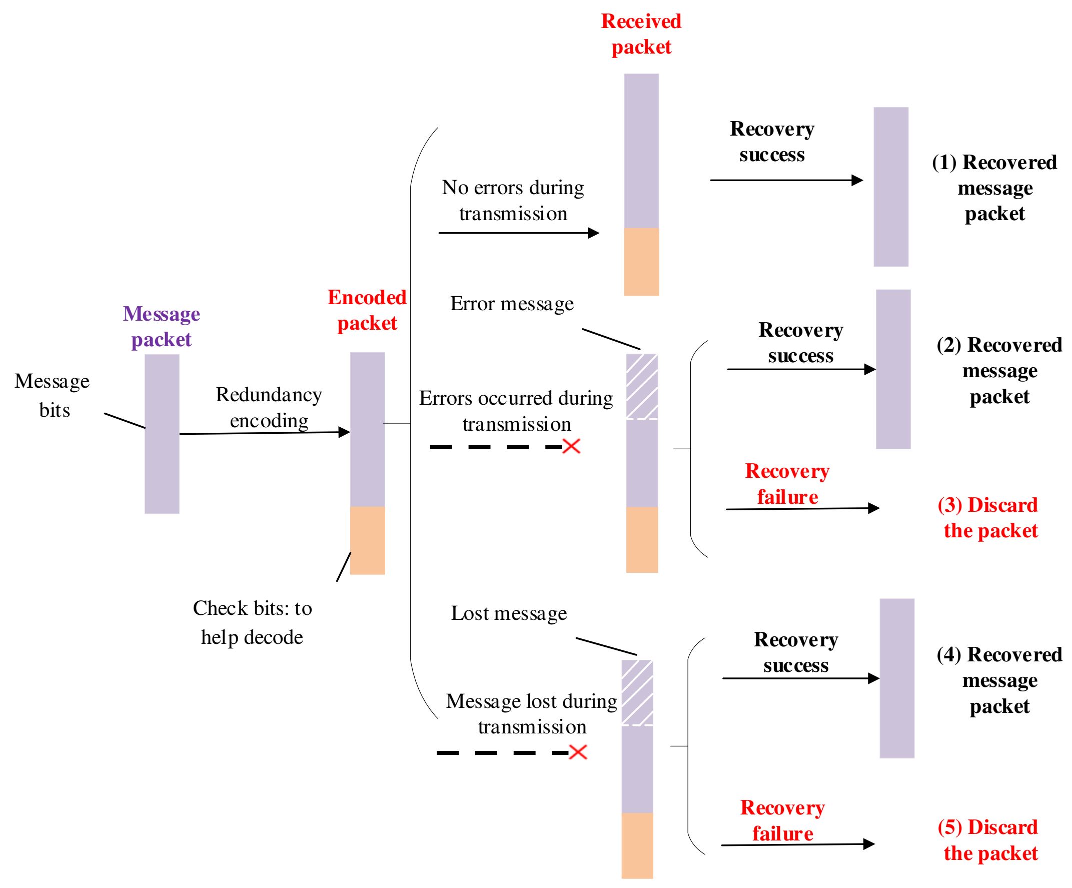


Fig. 3. The flowchart of the FEC mechanism. The figure shows the error processing and packet loss, demonstrating that in most cases, the FEC mechanism can recover data packets, and in other cases, it directly discards data packets.

As indicated in [8], in burst loss transmission channels, a receiver may not accumulate a sufficient number of packets in the FEC-encoded blocks, so it will be unable to reconstruct the complete original data, which yields to ineffective transmission in the FEC mechanism. Taghikhaki et al. [9] proposed a reliable data transmission protocol (RAFEC*), which can evaluate the link quality by counting the numbers of failures and successes in a certain period. The main advantage of this protocol is that it can adjust the error control mechanism automatically after several successes (or failures) rather than after only one success (or failure). Hence, the obvious benefit of this mechanism is that it can dynamically adjust the strength and complexity of the code according to the current requirements. However, this mechanism yields to the waste of wireless network resources and does not consider the reduction of packet length.

In [10], a new error control mechanism with the adaptive-performance sender and receiver was proposed. This mechanism takes full advantage of the characteristics of the FEC, allowing different FEC methods to be used in different network parts. Therefore, this mechanism not only provides flexible support to wireless networks but also reduces the resource consumption of wireless networks. Furthermore, this mechanism chooses between iterative and non-iterative codes according to the particular requirements, thus providing larger flexibility and reducing coding redundancy. In [11], a new adaptive error control (AEC) algorithm was proposed. The AEC algorithm adaptively changes error correction code (ECC) based on the channel behavior in recent previous transmissions. Simulation results showed that the AEC algorithm achieved better error correction performances than the traditional techniques, but network resources were wasted due to the existence of redundant bits.

In [12], an error correction scheme for burst-noise channels, which includes several procedures such as segmentation, error detection, and assessment, was proposed. The experimental results showed that the proposed scheme achieved the recovery ratio of over 40% for corrupted packets in practice. In [13], a linear error detection and correction scheme for wireless networks was proposed. This scheme improves network performance in terms of the bit error rate and throughput. However, the reduction of coding redundancy has not been considered in [12,13].

In [14], an adaptive cross-layer error control protocol for wireless networks was proposed. This protocol achieves significant improvements over the existing error control schemes by dynamically determining suitable redundancy for the source packet in the wireless link layer, but the error correction ability does not show a significant advantage. In [15], in order to solve the problem of 4-bit bursts, two effective solutions were presented. One of them uses two interleaved single and double adjacent error correction codes, which can reduce the decoding complexity and delay at the cost of having more check bits.

In [16], an application layer FEC method for wireless network communication was proposed. The adaptability and efficiency of FEC technology in wireless networks have been significantly improved by constantly altering the redundancy in FEC blocks and attaching the most appropriate redundancy amount. It was inadequate to cope with the drastic changes in network status. Similarly, in [17], another application layer FEC method is proposed based on small block low-density parity-check (LDPC) codes to protect real-time packetized multimedia streams in bursty channels. Although the scheme considers the wireless channel characters, it can only be used at the application layer for video file.

In [18], a modified FEC mechanism that uses the interleaving-coding-sequential-transmission (ICST) FEC, which rearranges the FEC coding sequence in an interleaved manner, and transmits packets sequentially according to their generation order, was introduced. This mechanism reduces both the burst loss impact and the delay time. In [19], the authors proposed an interleaved coding mechanism for wireless channel, it consider the impact of difference packet size, but ignore the wireless channel characters. Therefore in both [18] and [19], the reduction of packet redundancy has not been considered.

Hybrid ARQ and FEC

The ARQ and FEC mechanisms are two basic error control techniques, and both of them have advantages and disadvantages. The principle of the ARQ is simple, but the usage rate of the communication channel is low due to the need for retransmission. In contrast, the FEC encoding is complex, and a larger bandwidth is required due to the appending bits. However, the hybrid ARQ&FEC combines the advantages of both methods to improve the overall performance. Specifically, in the hybrid ARQ&FEC, the FEC corrects errors, and the ARQ compensates for errors that the FEC cannot correct, achieving the lowest bit error rate. The workflow of a hybrid ARQ&FEC error correction method is presented in Figure 4.

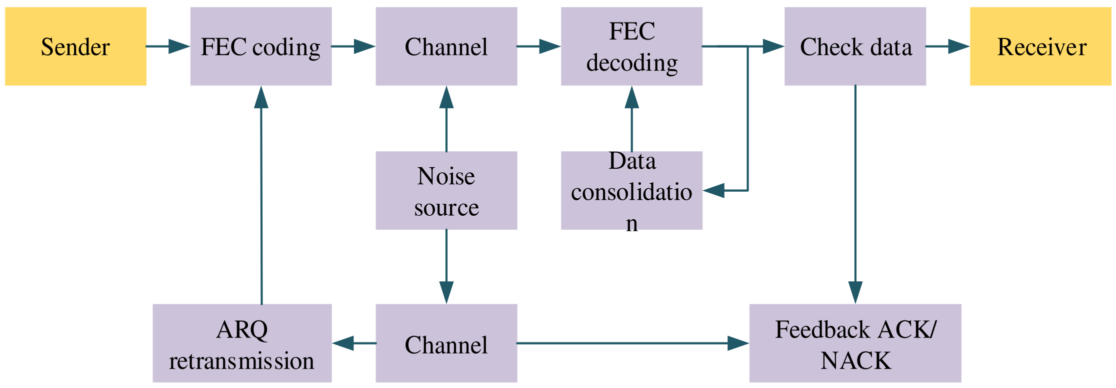


Fig. 4. The workflow of a hybrid ARQ&FEC error correction mechanism

In [20], based on the IEEE802.15.4 protocol, several ARQ and FEC energy efficiency mechanisms were modeled, and an error control scheme that adaptively selects ARQ or FEC based on the received signal strength of a wireless link is proposed. In this scheme, the ARQ mechanism is used when the channel quality is good, and the FEC technology is used when the channel quality is poor. Hence, this scheme can adapt to different channel qualities by combining the advantages of ARQ and FEC mechanisms. This scheme jointly considers the requirements for low transmission loss and different qualities of links between nodes in order to maximize energy efficiency. The experimental results show that this scheme achieves better performance in error correction than standard ARQ and FEC mechanisms separately. However, since the FEC method is used, most messages can be checked successfully, which causes high coding redundancy and a waste of network resources.

In [21], a scheme that extends the notion of current hybrid ARQ was proposed. Unlike other hybrid ARQ&FEC mechanisms, in this mechanism, when packet retransmission is required, the retransmission scheme jointly embeds the original information from the first packet and additional information from the second packet, providing a significant increase in coding gain of the first packet. The main disadvantage of this mechanism is a large usage of network resources due to the increased length of retransmitted packets.

As presented above, many schemes have been proposed to solve the problem of unreliable transmission. ~~However, only a few works tried to improve the error correction ability and reduce the coding redundancy simultaneously.~~ However , most of them does not consider the packet loss character in the wireless networks, thus they can not guarantee low redundancy and high error correction ability simultaneously.

Although the error correction ability has been enhanced in the existing work, a large amount of network resources have been wasted in the transmission process due to inappropriate check bits. In order to address this problem, this paper proposes a mechanism that can reduce the length checking redundancy bits by one-third using the XOR operation and thus improve the network utilization significantly. Besides, though most of the mechanisms or algorithms have tried to reduce the checking redundancy by some operations, such as adjusting redundant bits dynamically, they have not been focused on the improvement of error correction ability. In order to overcome this problem, the proposed work considers the improvement of error correction ability using the relationship between adjacent packets to provide checking help.

With the aim to solve the problem of burst errors, which is the major cause of transmission unreliability, the RS code is adopted in the proposed mechanism. In this way, burst errors are checked and corrected more easily and effectively. The goal of this study is to save network resources and improve error correction ability simultaneously and thus enhance network throughput.

3. METHODS

In this section, an error control mechanism based on RS codes is proposed to deal with burst errors in wireless networks. The basic idea of the proposed mechanism is to construct a relationship between consecutive packets, which is named the cross-relationship. Using this relationship, the error correction ability of packets can be increased, while packet length can be reduced. In addition, packets that cannot be checked successfully at the improved error correction ability will be retransmitted. The proposed mechanism can be divided into three parts: RS re-encoding, error checking, and packet retransmission, which are explained in the following.

A. RS re-encoding

In this section, RS coding and the re-encoding method based on RS code are described.

1. RS coding

The RS code is a special non-binary error correction code. In [22], the author evaluate the performance of RS code in some wireless channels, and point out some design factors to optimize the performance. This further illustrate that we need to design RS based error control mechanism according to our practical environment. For a positive integer S, the corresponding base-q BCH code with a code length

$$n = q^{S-1}$$

is constructed, where q denotes a power of a prime number. The base-q BCH code is called the RS code, and

$$n = q - 1,$$

$$s = 1,$$

and $q > 2$. When

$$q = 2^m \ (m > 1),$$

the binary RS code constructed over $GF(2^m)$ [23] can be used to correct burst errors, which is the most common RS code.

The RS codes are often expressed as $RS(n,k)$, where *n* denotes the length of a data packet, and *k* denotes the length of the message segment. In addition,

$$n - k = 3t,$$

where 3*t* represents the whole length of the checking segment. Furthermore, assume the message segment is expressed as $\{x_1 \ldots x_k\}$, then the RS code can be expressed as

$$\{x_1, x_2, \ldots, x_k, x_{k+1}, \ldots x_{k+\mathrm{t}}, x_{k+\mathrm{t}+1}, \ldots x_{k+2t}, x_{k+2t+1}, \ldots, x_{k+3t}\},$$

where the message segment is from the first bit to the *k*th bit. The proposed mechanism defines that the message segment can be expressed as $(x_1, \ldots, x_k)$. In addition, bits from (*k* + 1) to (*k* + 3t) represent checking bits, and they can be expressed as $(x_{k+1}, \ldots, x_{k+t})$, $(x_{k+t+1}, \ldots, x_{k+2t})$, and $(x_{k+2t+1}, \ldots, x_{k+3t})$, respectively. The coding structure of an RS coding packet is presented in Figure 5.

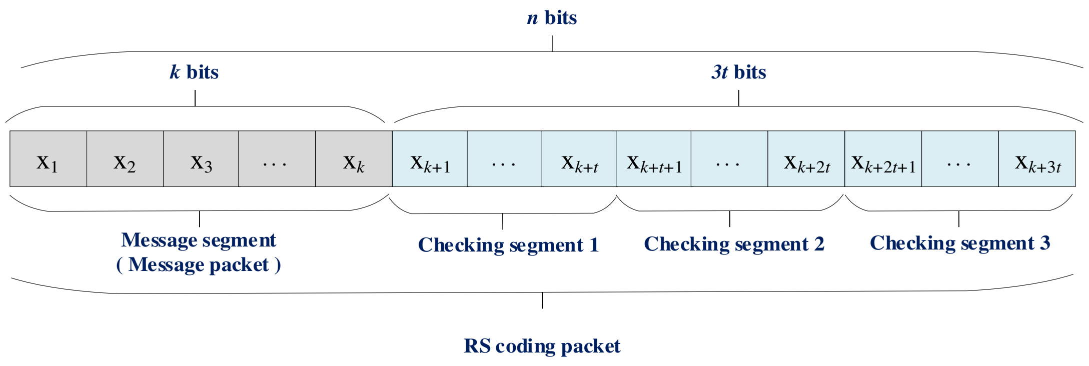


Fig. 5. The coding structure of an RS coding packet

2. Re-encoding

Re-encoding is the process of recombining the RS checking code by XOR operation. During the re-encoding process, the cross-relationship is constructed by performing XOR operation on adjacent packets. The re-encoding process is as follows.

(1) Message segment and the first RS checking segment remain the same; copy the k-bit message segment and the first t-bit checking segment to the beginning of a packet to be sent.

(2) The XOR operation is performed on the second RS checking segment of the current packet and the third RS checking segment of the previous packet, appending the result to the first checking segment as the second checking segment of the sending packet.

(3) The XOR operation is performed on the third RS checking segment of the current packet and the second RS checking segment of the next packet, and the result is added to the second RS checking segment of the next packet, which is then sent.

In this way, it can be guaranteed that the information of the current packet is contained in the next packet, and the message segment with only two checking segments containing $(k + 2t)$ bits in total represents the data to be sent. As a result, the packet length is reduced, and the cross-relationship is constructed. The re-encoding process of the RS code is presented in Figure 6.

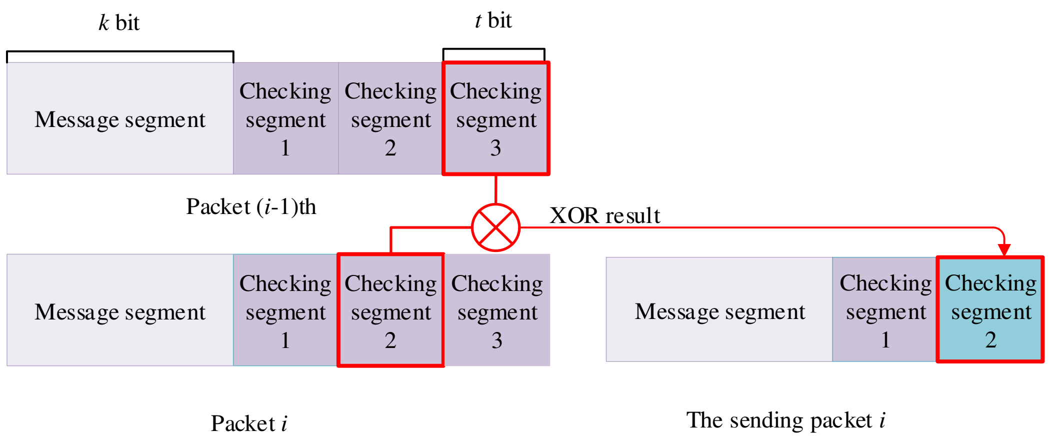


Fig. 6. The re-encoding process of the RS code

The sending data can be expressed as follows:

$$\{x_{i,1}, x_{i,2}, \ldots x_{i,k}, x_{i,k+1}, \ldots, x_{i,k+t}, x_{i,k+t+1} {}^\wedge x_{i-1,k+2t+1}, \ldots, x_{i,k+2t} {}^\wedge x_{i-1,k+3t}\},$$

where the first index of x represents the sequence number of a data packet, and the second index denotes the number of RS code position of the data packet; for instance, xi,2 refers to the second bit of packet i. The coding structure of a re-encoding packet is presented in Figure 7.

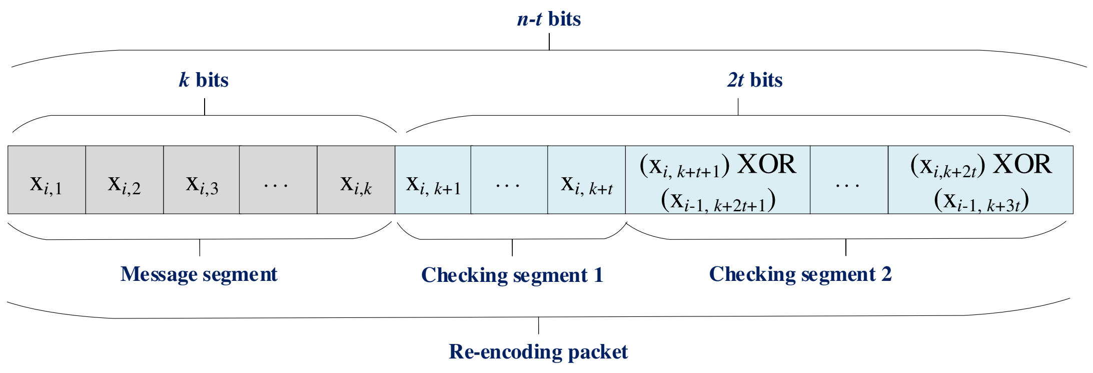


Fig. 7. The coding structure of a re-encoding packet

The structures of consecutive RS coding packets and consecutive re-encoding packets are displayed in Figure 8. In Figure 8, it can be seen that the checking segment 3 does not need to be sent, but the checking segment 3 is XOR-subjected with the checking segment 2 of the next packet, and the XOR result becomes the new checking segment 2 that is then sent. Therefore, a re-encoded packet has three parts to be sent: message segment, checking segment 1, and the result of the XOR operation of checking segments 2 and 3.

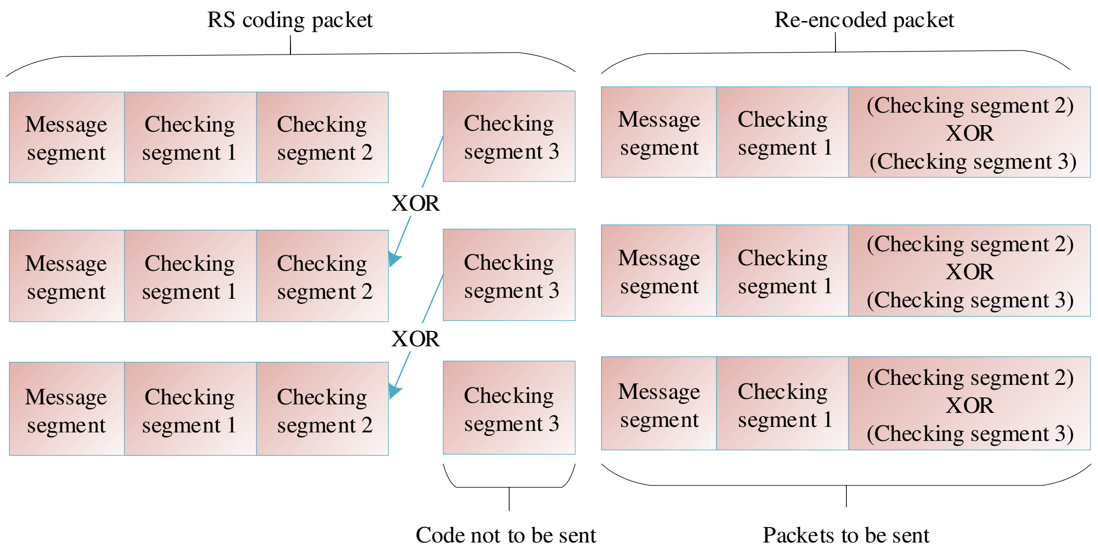


Fig. 8. The structures of RS coding packet and re-encoded packet

Therefore, a message segment needs to go through two steps from the initial data packet to the packet to be sent. The first step is to generate an RS-coding packet by the RS coding process, and in the second step, the generated RS-coding packet undergoes the re-encoding process to generate a re-encoded packet. The data structure of a re-encoding packet is presented in Figure 9.

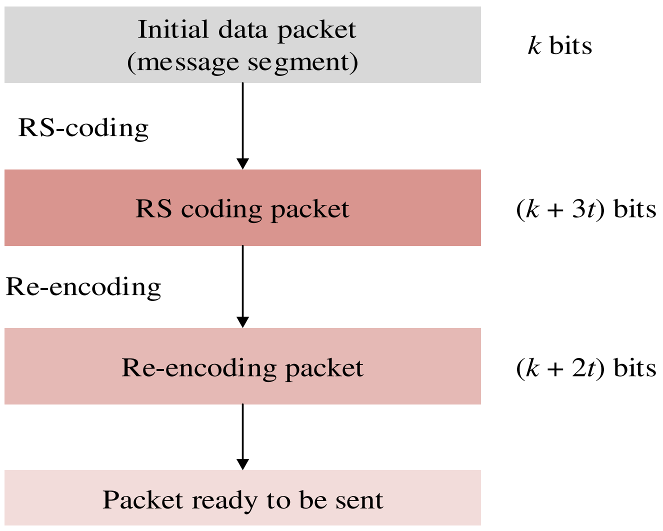


Fig. 9. The process of generating packets to be sent

B. Error Checking

Based on the re-encoding mechanism performed at the sender side, a checking mechanism named the cross-checking mechanism is performed at the receiver side. The cross-checking means that packets are checked using the cross-relationship. Specifically, when a packet is received at the receiver, the checking segment 1 is used to check the current packet. The cross-checking method is employed to help to check the packet only when the current packet cannot be recovered successfully by its own error correction ability of the checking segment 1.

In order to guarantee the cross-checking performance, the receiver buffers the newly received packet, which is then used in the packet checking process of the next packet which may need the checking assistance.Moreover, there is an intact t-bit calculated checking segment in each packet, which means that each packet has the $t/2$ error correction ability to correct errors. If two adjacent packets of a particular packet have been checked successfully, then the packet can have up to $3t$-bit checking segments.

Assume packet $(i - 1)$ has been successfully checked and stored in the receiver buffer. The next step is to receive packet i and check it by its own error correction ability. The cross-checking process after the reception of packet *i* is discussed in the following.

1. The packet *i* can be checked successfully by its own error correction ability.

Store packet *i* into the receive buffer and clear all packets whose sequence numbers are lower than *i* from the buffer.

2. The packet *i* cannot be checked successfully by its own error correction ability.

The packet *i* is checked using the increased error correction ability provided by the previous packet. The error checking process is as follows. First, the message segment of packet (*i* – 1) is encoded based on the generated RS code, and the check bits are divided into three *t*-bit checking segments. After that, XOR operation is conducted on the third segment of packet (*i* – 1) and the second segment of packet *i*, obtaining the second checking segment of packet *i*. At this time, packet *i* has 2*t*-bit error correction ability. The error checking process with the improved error correction ability of packet *i* is presented in Figure 10, where it can be seen that with the help of packet (*i* – 1), the error correction ability of packet *i* is increased from *t*-bit to 2*t*-bit.

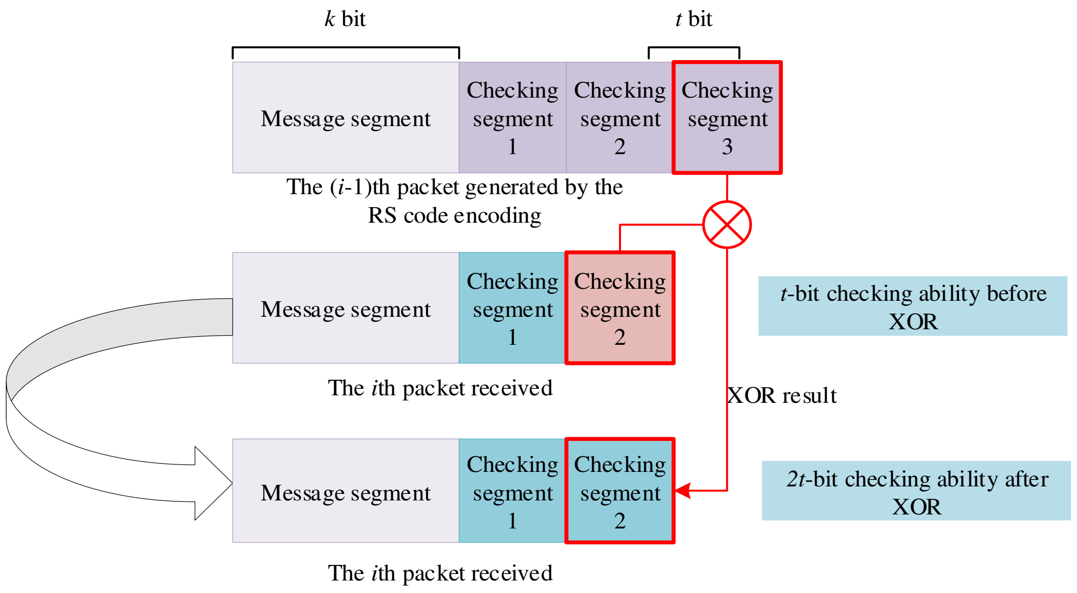


Fig. 10. The error checking process with the improved error correction ability of a packet

In the following, checking results of packet *i* obtained with the help of packet (*i* - 1) are discussed.

(1) The packet *i* can be checked successfully with the help of packet (*i* - 1)

This case is the same as Case 1 provided above. Therefore, store the packet *i* into the receiver buffer and clear packet (*i* - 1) from the buffer.

(2) The packet *i* cannot be checked successfully even with the help of packet (*i* - 1)

In this case, the receiver waits for the results of the next packet. The flowchart of obtaining the results of packet *i* with the help of packet (*i* - 1) is presented in Figure 11.

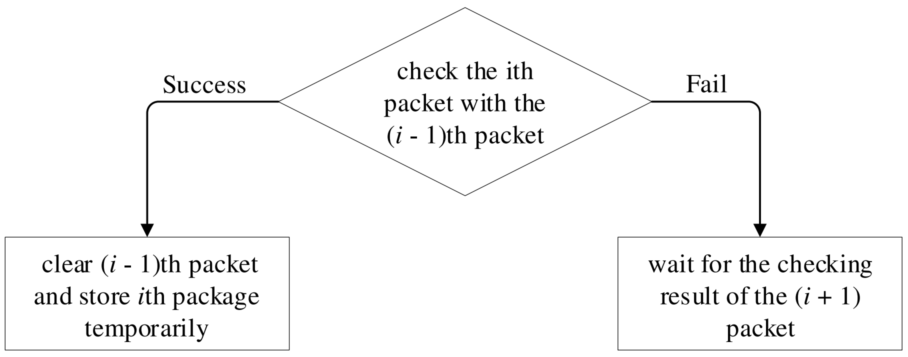

check the ith packet with the (i - 1)th packet
Success
Fail
clear (i - 1)th packet and store ith package temporarily
wait for the checking result of the (i + 1) packet

Fig. 11. The flowchart of the checking results of packet *i* with the help of packet (*i* – 1)

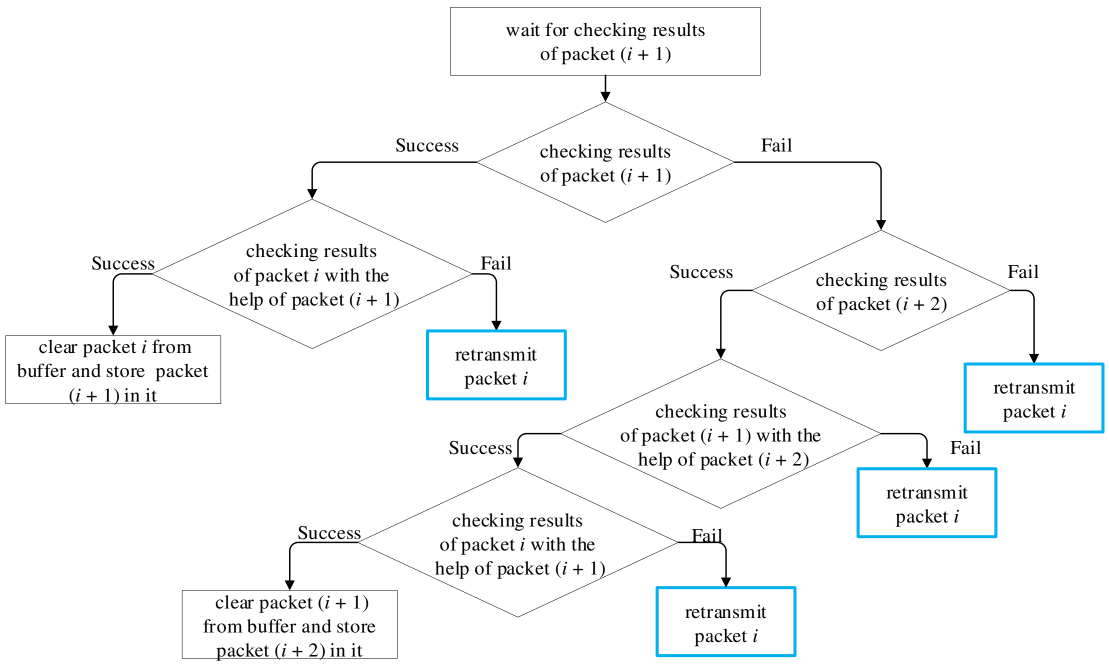
The checking results of packet (*i* + 1) are discussed below.

① The packet (*i* + 1) can be checked successfully by its own error correction ability

In this case, the first step is re-encoding the message segment of packet (*i* + 1) according to the generated RS code, obtaining the calculated check bits of packet (*i* + 1). Next, these check bits are divided into three *t*-bit segments. After that, XOR operation is performed on the calculated and received second checking segments of packet (*i* + 1), obtaining a *t*-bit checking segment as the third checking segment of packet *i*. Then, packet *i* is checked with the help of the newly obtained *t*-bit error correction ability. If the check process is successful, the *i*th packet will be cleared from the receiver buffer, while the (i + 1)th packet will be stored in the buffer; otherwise, the *i*th packet will be retransmitted. The retransmission mechanism will be introduced in Section C: Packet Retransmission.

② The packet (i + 1) cannot be checked successfully by its own error correction ability

In this case, the mechanism waits for the (i + 2)th packet. Similarly, if packet (i + 2) can be successfully checked, packet (i + 1) is checked with the help of packet (i + 2). If packet (i + 1) can be checked successfully, this case will become case ①; otherwise, packet i will be retransmitted. The checking process of cases ① and ② is illustrated in Figure 12.



wait for checking results of packet (i + 1)
checking results of packet (i + 1)
Success
Fail
checking results of packet i with the help of packet (i + 1)
Success
Fail
clear packet i from buffer and store packet (i + 1) in it
retransmit packet i
checking results of packet (i + 2)
Success
Fail
retransmit packet i
checking results of packet (i + 1) with the help of packet (i + 2)
Success
Fail
retransmit packet i
checking results of packet i with the help of packet (i + 1)
Success
Fail
clear packet (i + 1) from buffer and store packet (i + 2) in it
retransmit packet i

Fig. 12. The flowchart of the packet checking process

C. Packet Retransmission

If packet i cannot be checked successfully, even with the help of two adjacent packets, it will be retransmitted. The sender side retransmits a packet until either the packet can be checked successfully or the maximum number of retransmissions is reached.

In the proposed mechanism, the maximum retransmission number is set to three in order to prevent excessive network resources waste under the condition of bad channel status. Compared to the maximum number of retransmissions of the traditional ARQ mechanism, which is equal to seven, the proposed mechanism provides a stronger checking capability by adopting the cross-checking mechanism. Therefore, the transmission channel status is considered very bad after three retransmission failures.

A packet can have the error correction ability of up to $3t$, of which $2t$ is from two adjacent packets. As shown in Table 1, there are three cases in which packet i needs to be retransmitted. Assume packet (i - 1) can be successfully checked, and packet i cannot be successfully checked even with the help of packet (i - 1). Three cases are discussed below.

Table 1 Three cases in which packet *i* needs to be retransmitted.

	Case 1	Case 2	Case 3
Packet (i - 1)	√	√	√
Packet i	X	X	X
Packet (i + 1)	√	X	X
Packet (i + 2)		√	X

Note: X represents decoding failure, and √ represents decoding success.

Case 1: The packet (i + 1) can be successfully checked, but packet i cannot be successfully checked even with the help of packet (i + 1). In this case, packet i needs to be retransmitted, and it does not need to wait for the checking result of packet (i + 2) because packet i cannot be successfully checked even with the error correction ability of $3t$.

Case 2:The packet (i + 1) cannot be successfully checked, but packet (i + 2) can be successfully checked. However, packet $(i + 1)$ cannot be successfully checked even with the help of packet $(i + 2)$. In this case, packet i needs to be retransmitted because its error correction ability of $2t$ cannot be increased, and packet i cannot be successfully checked with its current error correction ability of $2t$.

Case 3: Packets $(i + 1)$ and $(i + 2)$ cannot be successfully checked. This paper stipulates that when there are three consecutive packets that cannot be successfully checked, the first packet needs to be retransmitted because the error correction ability of the first packet of $2t$ cannot be increased by other packets.

For each retransmission, there are three possible situations: (1) packet i can be successfully checked by its own error correction ability after retransmission; (2) packet i cannot be successfully checked by its own error correction ability after retransmission; or (3) the maximum number of retransmissions is reached. Each situation is discussed in the following.

(1) Packet i can be successfully checked by its own error correction ability after retransmission.

In this situation, packet i can be successfully checked by its own error correction ability after retransmission, so the subsequent checking processes of Cases 1–3 in Table 1 are as follows.

Case 1: Since packet $(i + 1)$ has been successfully checked, then only the message of packet $(i + 1)$ will be stored in the receiver buffer, it is set that

$$i = i + 2,$$

and the retransmission process returns to the transmission process. The subsequent checking processes of Case 1 after successful self-check of packet i is shown in Figure 13.

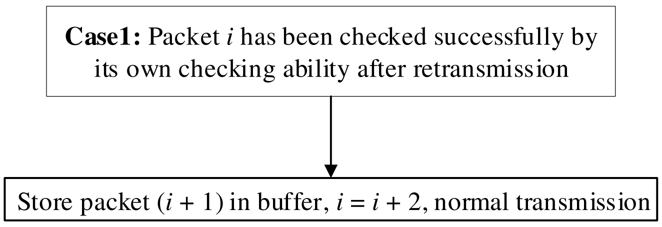

Case1: Packet *i* has been checked successfully by its own checking ability after retransmission
Store packet (*i* + 1) in buffer, *i* = *i* + 2, normal transmission

Fig. 13. Subsequent checking processes of Case 1 after successful self-check of packet *i*

Case 2: Check packet (i + 1) with the help of packet i. If the checking process is successful since the packet (i + 2) has been successfully checked, then only the message of packet (i + 2) will be stored in the receiver buffer, it is set that

$$i = i + 3,$$

and the retransmission process returns to the transmission process; otherwise, the sender side retransmits packet (i + 1), and the receiver side checks it. The subsequent checking processes of Case 2 after successful self-check of packet i is shown in Figure 14.

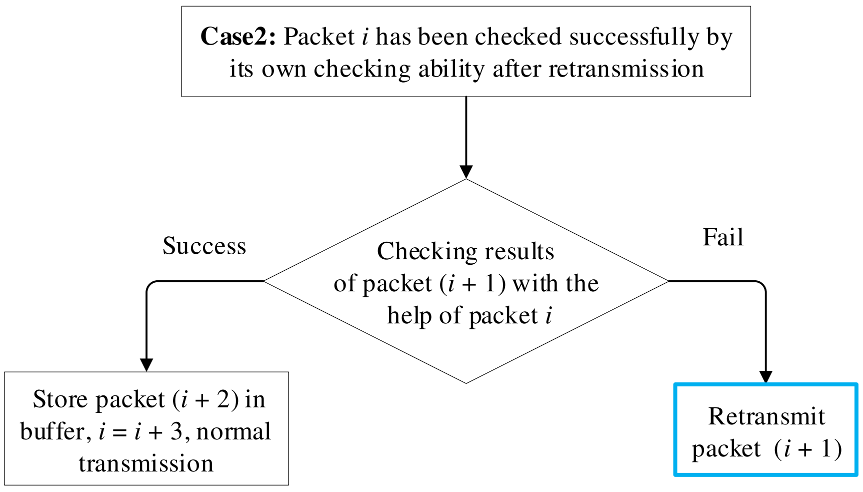

Case2: Packet *i* has been checked successfully by its own checking ability after retransmission
Success
Checking results of packet (*i* + 1) with the help of packet *i*
Fail
Store packet (*i* + 2) in buffer, *i* = *i* + 3, normal transmission
Retransmit packet (*i* + 1)

Fig. 14. Subsequent checking processes of Case 2 after successful self-check of packet *i*

Case 3: Check packet (i + 1) with the help of packet i. If the checking process is successful, check packet (i + 2) with the help of packet (i + 1). If the checking process is successful, then only the message of packet (i + 2) will be stored in the receiver buffer, it is set that

$$i = i + 3,$$

and the retransmission process returns to the transmission process; otherwise, the sender side retransmits packet (i + 2), and the receiver side checks it. However, if packet (i + 1) cannot be checked successfully with the help of packet i, the sender side retransmits packet (i + 1). The subsequent checking processes of Case 3 after successful self-check of packet i is shown in Figure 15.

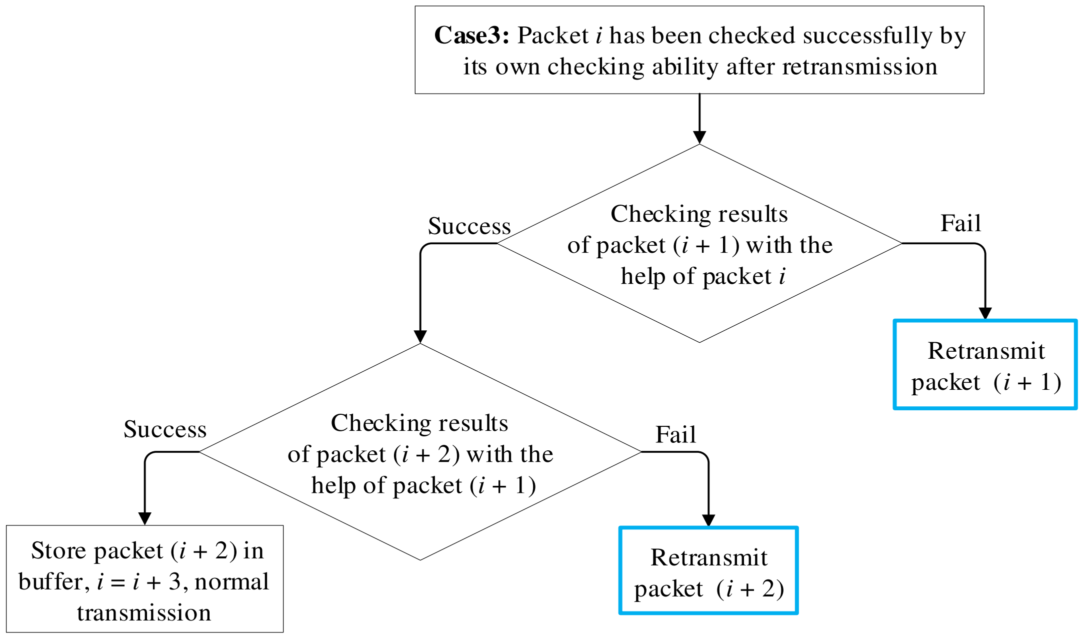

Case3: Packet i has been checked successfully by its own checking ability after retransmission
Success
Checking results of packet (i + 1) with the help of packet i
Fail
Retransmit packet (i + 1)
Success
Checking results of packet (i + 2) with the help of packet (i + 1)
Fail
Store packet (i + 2) in buffer, i = i + 3, normal transmission
Retransmit packet (i + 2)

Fig. 15. Subsequent checking processes of Case 3 after successful self-check of packet i

(2) Packet *i* cannot be successfully checked by its own error correction ability after retransmission.

In this situation, if packet i cannot be successfully checked by its own error correction ability after retransmission, the subsequent checking processes of Cases 1–3 are as follows.

Case 1: check retransmitted packet i with the help of packet (i - 1) and packet (i + 1). If the checking process is successful, then only the message of packet (i + 1) will be stored in the receiver buffer, it is set that

$$i = i + 2,$$

and the retransmission process returns to the transmission process; otherwise, continue to retransmit packet i.

Case 2: check retransmitted packet i with the help of packet (i - 1). If the checking process is successful, check packet (i + 1) with the help of packet i and packet (i + 2); otherwise, the sender side retransmits packet i.

Case 3: check retransmitted packet i with the help of packet (i - 1). If the checking process is successful, check packet (i + 1) with the help of packet i; otherwise, the sender side retransmits packet i.

(3) The maximum number of retransmissions is reached

When the maximum number of retransmissions has been reached, and the packet has not been successfully checked, the packet will be discarded. The specific processes of Cases 1–3 are as follows.

Case 1: The sender side discards packet i, and receiver buffer discards packet (i + 1), it is set that

$$i = i + 1,$$

and the retransmission process returns to the transmission process.

Case 2: The sender side discards packet i, and receiver buffer discards packets (i + 1) and (i +

2), it is set that

$$i = i + 1,$$

and the retransmission process returns to the transmission process.

Case 3: This case is the same as Case 2.

D. Theoretical Analysis

In this part, a simple wireless channel model will be constructed for analyzing the effect of our mechanism. Packet retransmission if of great influence on the network performance, so we will compare the packet retransmission probability of RS code and our improved RS re-coding method with the same number of check bits.

In the wireless channel model, we suppose the channel will be in one of the two states: bad state (B) and good state (G)[17], as shown in Figure 16. The bit error rate is low in good state, so both the RS code and our improved RS code can correct the error bits. While in bad state, the bit error rate is high, the number of error bits may exceed the ability of the check bits, so the packet needs to be retransmitted.

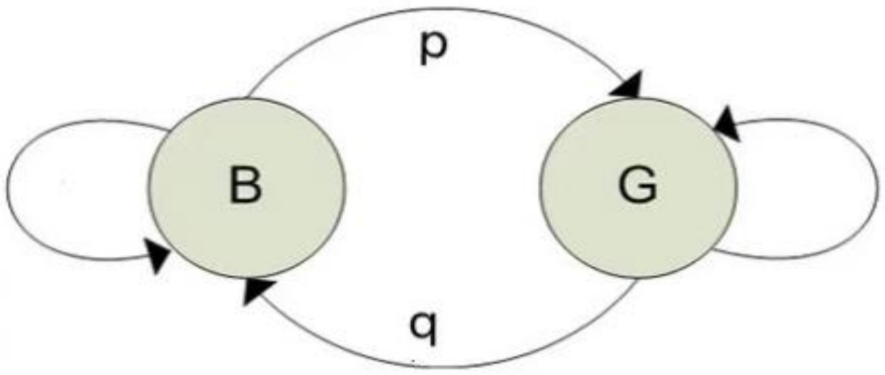


Fig.16. Wireless channel model

Based on this model, we can get the steady state probability of bad state($\pi(B)$):

$$\pi(B) = \frac{p}{q+q},$$

the probability of two continuous bad states($\pi^2(\mathrm{B})$):

$$\pi^2(\mathrm{B}) = \frac{(1-p)q}{p+q} ,$$

and the probability of two continuous bad states($\pi^3(B)$):

$$\pi^3(B) = \frac{(1-p)^2 q}{p+q} ,$$

where p, q are the transition probabilities.

The error correction ability of RS code with 2t check bits is t. This means when the number of error bits is lager than t, the packet can not be corrected and needs to be retransmitted. So its packet retransmission probability is

$$\pi(B) \cdot P(ne > t)$$

In our improved RS $_{\mathrm{r}}$e-coding mechanism, the packet retransmission is corresponding to the three cases shown in Table1.

In case1, the packet will be retransmitted when the number of error bits (ne) is lager than $\frac{3}{2}\mathrm{t}$ .It's probability is:

$$\pi(B) \cdot P\left(ne > \frac{3}{2}t\right)$$

In case2, retransmission means the number of error bits in two continuous are both lager than t . It's probability is:

$$\pi^2(B) \cdot P(ne > t)$$

In case3, retransmission happens when the number of error bits in one packet is lager than t , while for the other two followed packets are lager than $\frac{1}{2}\mathrm{t}$. It's probability is:

$$\pi^3(B) \cdot P\left(ne > \frac{1}{2}t\right)$$

So the packet retransmission probability of our mechanism is

We assume that when the channel is in a bad state, the number of error bits follows the distribution of $N(t, \sqrt{t})$ .The value of p and q depends on the practical network states, we let q equal to 0.1, t equal to 16, and p vary from 0.9 to 0.4, then the packet retransmission probability can be computed as shown in Figure 17.

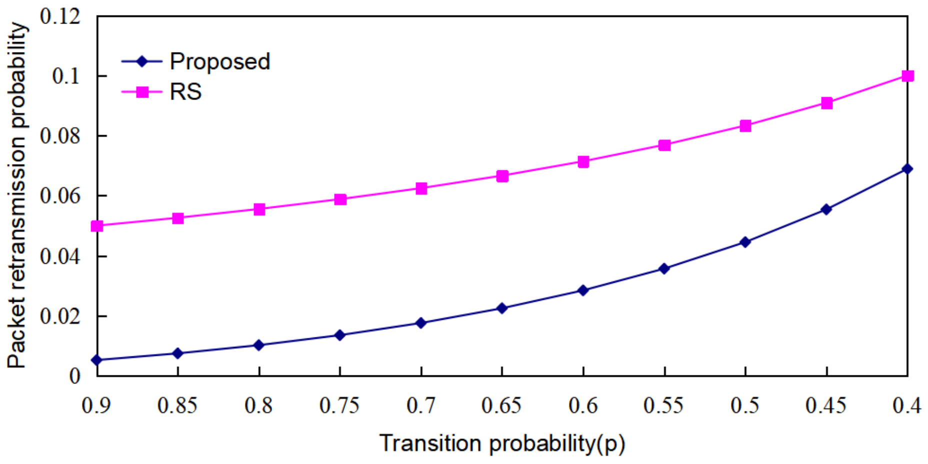


Fig.17. Packet retransmission probability

From Figure 17, one can see that our mechanism outperforms the RS code with smaller packet retransmission probability.

In this model, we only compare the performance with RS code. The error correction ability of CRC is weaker than RS code, so its packet retransmission probability should be much higher. We make some assumptions for the distribution of the number of error bits, this will not affect the comparison results, we can get similar results for other distribution.

4. Results and Discussion

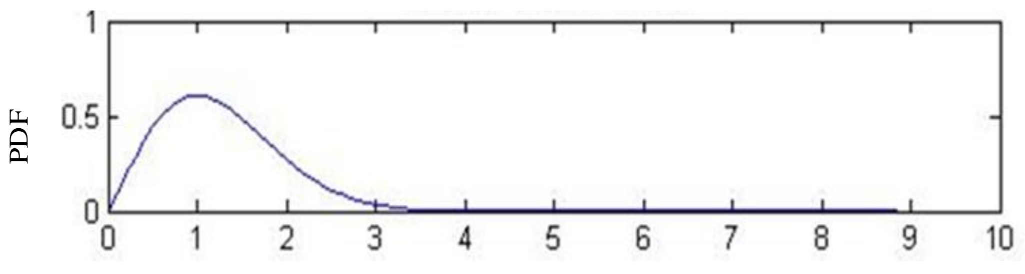
The performance of our mechanism was evaluated by the NS2 simulation software [24] using the IEEE 802.11 protocol. The proposed mechanism was ~~also~~ compared with the default error control mechanism, i.e., the Cyclic Redundancy Check(CRC) [25]. We also implemented the error control mechanism based on RS with $2t$ (RS_32) check bits and $3t$ check bits(RS_48) [22].

In the experiments, message segment k was set to 255 bits, and $(n - k)$ was 32 bits, this means t is set to 16. During the experiments, the sizes of the uplink and downlink buffers were the same. The following parameters were unchanged. At the link layer, the packet length was 255+32 bits, of which 32 bits were the checking bits. Furthermore, the wireless bandwidth was 11 Mbps, the wired bandwidth was 10 Mbps, the simulation time was 300 s, and the default buffer size was 100 packets. The experimental parameters are given in Table 2.

Table 2 Experimental parameters

Simulation parameter	Value	Simulation parameter	Value
Packet size	1500 bytes	Default buffer size	100 packets
Packet length	287 bits	Default simulation time	300 s
Number of message bits	255 bits	Wireless bandwidth	11 Mbps
Number of check bits	32 bits	Wired bandwidth	10 Mbps

First, the performance of the packet loss and the continuous number of lost packets of the proposed method were analyzed under different network topology scenarios. The probability distribution function (PDF) curve is shown in Figure~~16~~ 18.



The Number of continuous lost packets

Fig. 18. The PDF curve

As presented in Figure 16, in most cases, only one packet was lost, so the packet loss rate was very low. The possibility of losing three or more packets in the sequence was very low, so it was ignored. These results prove the rationality and feasibility of the proposed cross-checking method.

The number of retransmissions as a function of the bit error rate is shown in Figure~~17~~ 19. As displayed in Figure ~~17~~19, under good transmission channel conditions, the CRC method and our proposed method performed similarly, but when the bit error rate exceeded a value of 0.08, our mechanism was significantly superior to another scheme because our mechanism could reduce the number of transmissions by improving the error correction ability.

RS_48 performs the best while CRC32 is the worst, this is due to their error correction ability and the number of check bits. Our mechanism can get almost the same effect as RS_48, this illustrates the fact our cross-checking mechanism is effective. When the error rate is very high, RS 48 has a better effect. Because at this time, the channel becomes worse, and we cannot obtain the same result as RS 48 through cross-checking processes.

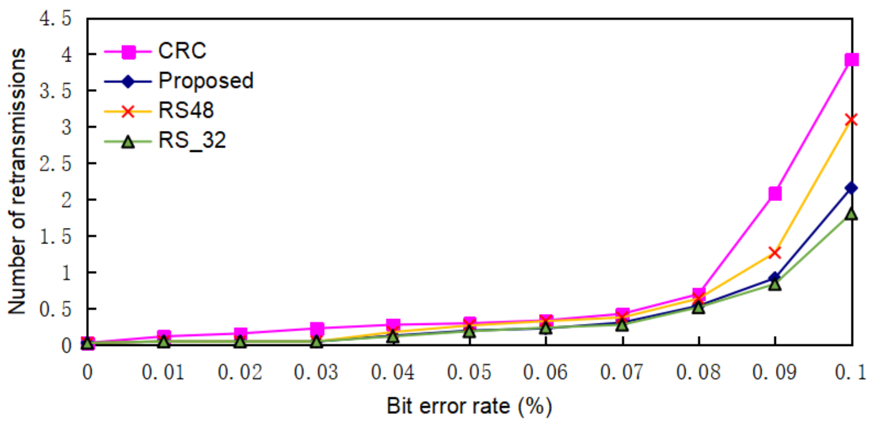


Fig. 19. The number of retransmissions dependence on the bit error rate

The network throughput of the proposed mechanism was analyzed by experiments under two different scenarios, ad-hoc scenario and multi-hop wireless scenarios, which are the most common scenarios.

In the first experiment, the proposed scheme was compared with the CRC in the ad-hoc wireless network, where some of the nodes were placed randomly in a given domain, and the traffic pattern was random. The random networks containing from 50 nodes to 100 nodes were simulated using the protocol 802.11 and our proposed mechanism successively.

The throughput as a function of node number is presented in Figure20, where it can be seen that both mechanisms showed similar decreasing trends when the number of nodes increased. However, with the increase in the number of nodes, the transmission conflicts between nodes increased, so the bit error rate also increased. Therefore, the decreasing trend of simulation results indicated that the bit error rate increased when the density of the network increased.

When the bit error rate is low, the result of RS_32 and RS_48 is worse than that of CRC, because the computational complexity of RS code is high, and the redundancy of RS_48 will waste the bandwidth. However, the coding proposed in this paper only needs to use 16-bit check bit when the error rate is low, which speeds up the computation process. When the error rate is high, our algorithm can obtain the error correction performance close to that of RS48, so the network performance is improved. Its throughput performance was better by about 25% than the performance of the CRC.

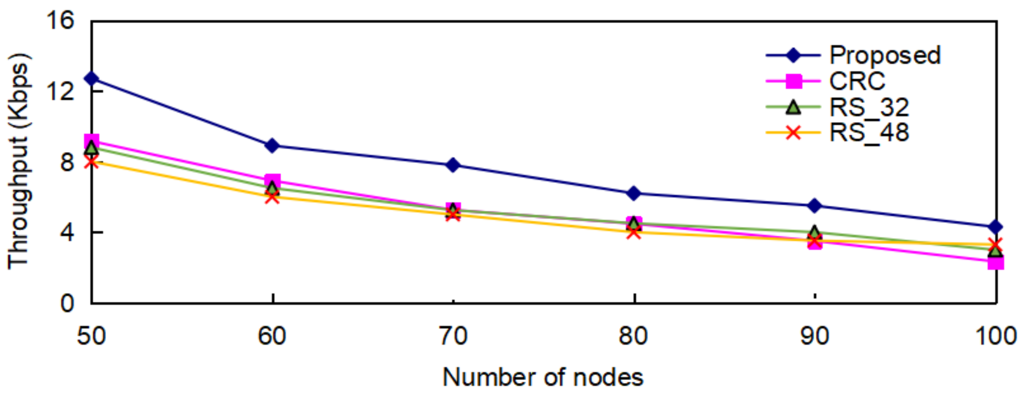


Fig. 20. The throughput

The percentage of failed packet checks as a function of node number is presented in Figure ~~19~~21, where it can be seen that ~~both~~ all the mechanisms showed similar increasing trends as the number of nodes increased. This was because, with the increase in the number of nodes, transmission conflicts between nodes increased, so the percentage of failed packet checking also increased.

In the case of checking failed packets, we also achieve similar results as RS_48. This further demonstrates the validity of our algorithm and the rationality of our design. The retransmission probability of RS 48 is better than that of our algorithm only when the number of nodes exceeds 90, i.e. when the packet loss rate increases due to collisions.

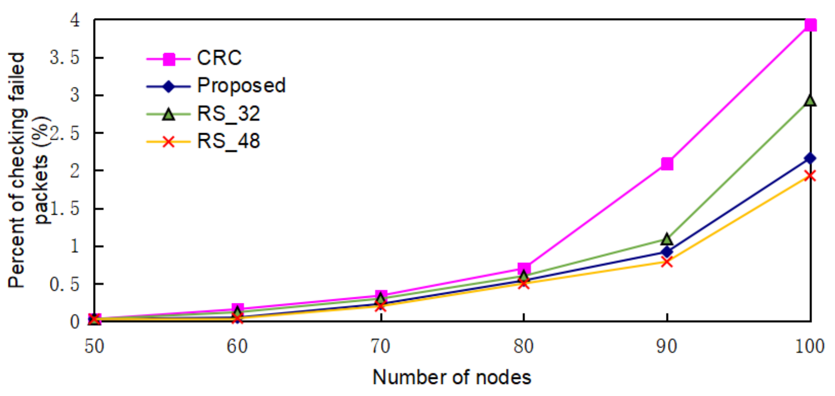


Fig. 21. The percentage of failed packet checks

In the second experiment, a chain topology with the cross-traffic was used, as shown in Figure~~20~~22. As presented in Figure 20, there were 10 senders (S1–S10) and 10 receivers (R1–R10) in total.

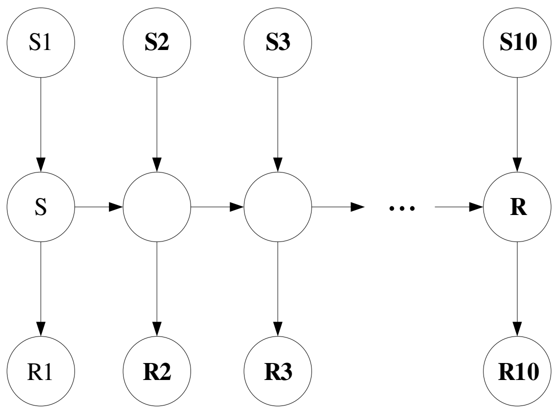


Fig. 22. The experimental topology

The throughput was analyzed as a function of the bit error rate. The obtained results are presented in Figure ~~21~~23 where it can be seen that when the bit error rate was low, the throughput performance of the proposed mechanism slightly differed from that of the traditional error control mechanism(CRC). However, with the increase in the bit error rate, the throughput of the proposed mechanism was better than that of the traditional mechanism. This was because, due to improved error correction ability, the proposed mechanism could reduce the number of retransmissions, thus improving the throughput capacity. As the bit error rate continued to increase, in some cases, the proposed mechanism could not correct the errors, resulting in a waste of error correction time, which further slightly reduced the error control performance of the proposed mechanism.

Compared with RS_32 and RS_48, they suffer form the high computational complexity and redundancy, so their throughput degrades especially when the bit error rate is low.

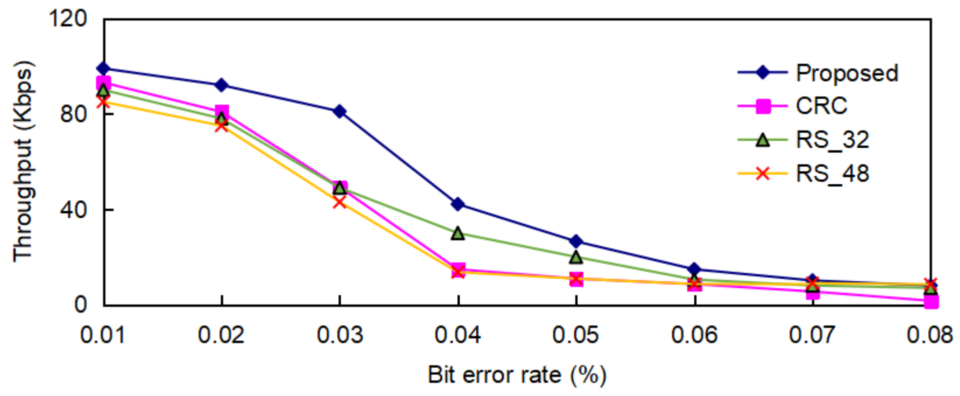


Fig.-23. The throughput

The percentage of failed packet checks as a function of the bit error rate is presented in Figure–24. As presented in Figure–24, the proposed mechanism achieved a lower percentage of failed packet checks than the CRC. When the bit error rate was about 0.01 and 0.08, the proposed mechanism and CRC had similar performances. When the bit error rate was between 0.02 and 0.07, the percentage of failed packet checks of the proposed mechanism was obviously lower than that of the CRC. The performance of our mechanism is similar as RS_48 and is better than RS_32.

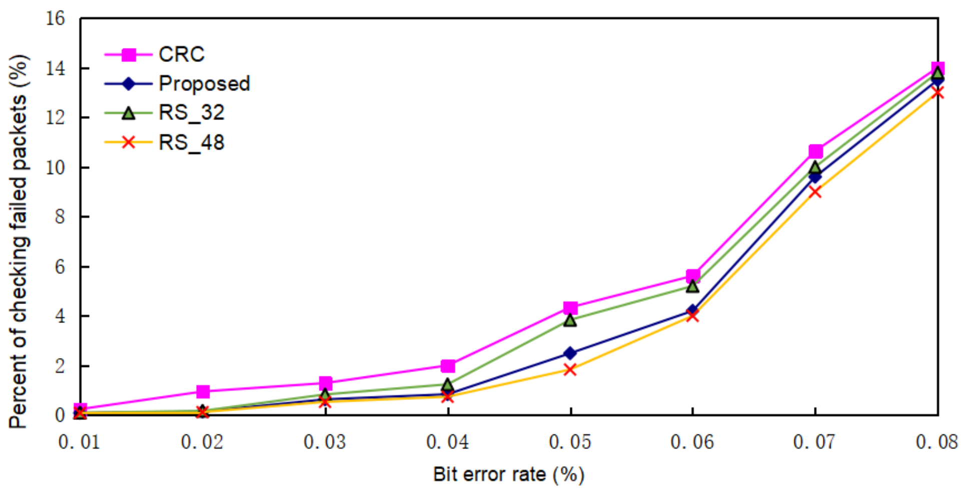


Fig. 24. The percentage of checking failed packets

According to the experimental results, when the bit error rate was between 0.02 and 0.07, the proposed mechanism performed better in terms of the throughput and percentage of failed packet checks. When the bit error rate exceeded the value of 0.08, the proposed mechanism required fewer retransmissions. Furthermore, with the increase in the number of nodes, the proposed mechanism could achieve a 25% better throughput performance than the CRC. Especially when the number of nodes exceeded the value of 80, the percentage of failed packet checks of the proposed mechanism was obviously lower than that of the CRC.

However, when the bit error rate was extremely high or extremely low, the proposed mechanism showed similar performance as the CRC in terms of the throughput, percentage of failed packet checks, and the number of retransmissions.

In summary, the decoding cost and redundancy of RS code will consume the network resource. Our algorithm provides three flexible decoding length (t, $2t$, $3t$) methods according to different networks, so it can well adapt to the network state, thus improving the network performance.

5. Conclusions

In this study, an error control mechanism based on the RS code is proposed, and its performance is compared with the traditional error control mechanism. The proposed cross-checking mechanism is based on the re-encoding of RS code, and it utilizes the cross-relationship of check bits in adjacent packets to increase the error correction ability. Compared with the traditional error correction mechanism, our solution can reduce the coding redundancy, thus saving the wireless network resources. In addition, our solution is particularly applicable for burst errors in wireless networks. The proposed mechanism was compared with the traditional mechanism by experiments using NS2 simulation software. The experimental results showed that the proposed mechanism showed obvious superiority over the traditional mechanism regarding both correction ability and throughput capability.

Although the proposed error control mechanism based on RS code for wireless networks can achieve good results in error control and correction, there are certain limitations. Namely, in this work, there is no time limit, but in practice, there is often a time limit in error control mechanisms. Therefore, the proposed mechanism would not be the best choice for error detection and correction for packets with a very high bit error rate. The proposed mechanism can be further improved, which will be part of our future research.

List of abbreviations

words	**abbreviations**
Reed-Solomon	RS
automatic repeat request	ARQ
forward error correction	FEC
cumulative feedback-based ARQ	CF ARQ
adaptive error control	AEC
error correction code	ECC
interleaving-coding-sequential-transmission	ICST
Bose–Chaudhuri–Hocquenghem codes	BCH

Declarations

Availability of data and materials:

Not applicable. Since we are doing simulation experiments, we do not have the project database and code. We can provide RS simulation code if the journal needs it.

Competing interests:

The authors declare that they have no competing interests.

Funding:

The work was supported by the National Key Research and Development Plan of China (No. 2018YFC0831405), the Natural Science Foundation of Tianjin (No. 18JCZDJC30700), the New Generation Artificial Intelligence Technology Major Project of Tianjin (No. 19ZXZNGX000801100).This work was supported in part by a Tianjin Science and Technology Project under grant No.19JCTPJC43300. These funding provide help in writing and polishing the manuscript.

Authors' contributions:

Xinyu Wang performed the data analyses and wrote the manuscript;

Kai Shi contributed to the conception of the study;

Jswang contributed significantly to the manuscript preparation;

Sheng Lin performed the mechanism analyses;

Guangping Xu perfermed the simulation experiment.

Fuqiang Qiao helped perform the analysis with constructive discussions.

Acknowledgements:

The authors would like to thank Jiayi Zhang for the help to the experiment. We would also like to thank the few anonymous reviewers.

Authors' information:

Xinyu Wang: Tianjin University of Technology, Computer Science and Engineering, Tianjin, China.

Kai Shi: Tianjin University of Technology, Computer Science and Engineering, Tianjin, China.

Jinsong Wang: Tianjin University of Technology, Computer Science and Engineering, Tianjin, China.

Sheng Lin: Tianjin University of Technology, Computer Science and Engineering, Tianjin, China.

Guangping Xu: Tianjin University of Technology, Computer Science and Engineering, Tianjin, China.

Fuqiang Qiao: Tianjin Sino-German University of Applied Sciences, College of Software and Communications, Tianjin, China.

REFERENCE

[1]S. Rayanchu, A. Mishra, D. Agrawal, S. Saha and S. Banerjee, "Diagnosing Wireless Packet Losses in 802.11: Separating Collision from Weak Signal," IEEE INFOCOM 2008 - The 27th Conference on Computer Communications, 2008, pp. 735-743.

[2]Frnda, J., Voznak, M. & Sevcik, L. Impact of packet loss and delay variation on the quality of real-time video streaming. Telecommun Syst 62, 2016,265–275.

[3] D. Malak, M. Médard and E. M. Yeh, "ARQ with Cumulative Feedback to Compensate for Burst Errors," 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 2018, pp. 1-6.

[4] Li L, Yuan B, Wang Z, et al. Unified architecture for Reed-Solomon decoder combined with burst-error correction[J]. IEEE transactions on very large scale integration (VLSI) systems, 2011, 20(7): 1346-1350.

[5] Wu Y. Novel burst error correction algorithms for Reed-Solomon codes[J]. IEEE transactions on information theory, 2012, 58(2): 519-529.

[6] P. Farkas et al., "New ARQ strategy inspired by Slepian-Wolf side information correlated source coding", IEEE EmergiTech, pp. 400-403, Aug 2016.

[7] N. Sakunnithimetha and U. Tuntoolavest, "An efficient new ARQ strategy for vector symbol decoding with performance in power line communications," 2017 International Electrical Engineering Congress (iEECON), Pattaya, 2017, pp. 1-4.

[8] P. Hsiao, C. Kuo, C. Shieh, W. Hwang, C. Ke and Y. Chen, "An analysis modeling of frame-level forward error correction for H.264/AVC over burst-loss channels," 2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE), Aalborg, 2014, pp. 1-5.

[9] Zahra Taghikhaki, Nirvana Meratnia, Paul Havinga, ” Protecting Informative Messages over Burst Error Channels in Chain-based Wireless Sensor Networks ” IEEE EmergiTech, pp. 400-403, Aug 2015.

[10] R. Kadel, K. Ahmed and A. Nepal, "Adaptive error control code implementation framework for software defined wireless sensor network (SDWSN)," 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), Melbourne, VIC, 2017, pp. 1-6.

[11] Yigit M, Boluk P S, Gungor V C. A new efficient error control algorithm for wireless sensor networks in smart grid[J]. Computer Standards & Interfaces, 2019, 63: 27-42.

[12] Wang Y, Li W, Lu S. The capability of error correction for burst-noise channels using error estimating code[C]//2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON). IEEE, 2016: 1-9.

[13] Alabady S A, Salleh M F M, Al-Turjman F. A novel approach of error detection and correction for efficient energy in wireless networks[J]. Multimedia Tools and Applications, 2019, 78(2): 1345-1373.

[14] Sarvi B, Rabiee H R, Mizanian K. An adaptive cross-layer error control protocol for wireless multimedia sensor networks[J]. Ad Hoc Networks, 2017, 56: 173-185.

[15] Li J, Xiao L, Reviriego P, et al. Efficient implementations of 4-bit burst error correction for memories[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2018, 65(12): 2037-2041.

[16] Tian H, Wu Y. An application layer forward error correction method of wireless network broadcasting communication for smart video surveillance system[J]. Journal of Ambient Intelligence and Humanized Computing, 2019: 1-9.

[17] F Casu, Cabrera J , F Jaureguizar, et al. A protection scheme for multimedia packet streams in bursty packet loss networks based on small block low-density parity-check codes, EURASIP Journal on Wireless Communications and Networking, 2015, 2015(1):187.

[18] Y. Weng, C. Shih, M. Lin and C. Shieh, "ICST: Low-delay forward error correction scheme with interleaved coding and sequential transmission," 2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), Nantou, 2016, pp. 1-2.

[19]H. Kasban and M. A. R. El-Tokhy M. A. M. El-Bendary, Interleaved Reed-Solomon Codes With Code Rate Switching over Wireless Communications Channels, International Journal of Information Technology and Computer Science, Vol.14, Issue,1, 2014.

[20] J. Chui and A. Chindapol, "Design of Cross-Packet Channel Coding with Low-Density Parity-Check Codes," 2007 IEEE Information Theory Workshop on Information Theory for Wireless Networks, Solstrand, 2007, pp. 1-5.

[21] B. Wang and Q. Zhang, "Study of performance comparison of satellite error correction codes for correcting big burst data errors," 2018 IEEE 3rd International Conference on Big Data Analysis (ICBDA), Shanghai, 2018, pp. 254-258.

[22]Ma, R.; Xing, L.; Wang, Y. Performance Analysis of Reed-Solomon Codes for Effective Use in Survivable Wireless Sensor Networks. Int. J. Math. Eng. Manag. Sci. 2020, 5, 13–28.

[23] A. Cilardo, "Fast Parallel GF(2^m) Polynomial Multiplication for All Degrees," in IEEE Transactions on Computers, vol. 62, no. 5, pp. 929-943, May 2013.

[24] https://www.scientific.net/paper-keyword/ns2-simulation

[25]IEEE standard，IEEE Std 802.11-2020，2020.