ROGUE PACKET DETECTION AND CORRECTION METHOD FOR DATA STORAGE DEVICE

Field of the Invention

The present invention pertains generally to data integrity in data recording/recovery devices, and more particularly to a method for detecting and correcting falsely detected "good" data packets recovered from a storage medium.

Related Patents

The present invention is related to co-pending U.S. patent application entitled "Method And Apparatus For Logically Rejecting Previously Recorded Track Residue From Magnetic Media", invented by McAuliffe et al., and having an internal docket number of 9086/103 and a serial number of 09/192,794, filed concurrently herewith on November 16, 1998, and co-pending U.S. patent application entitled "A Method Of Reacquiring Clock Synchronization On A Non-Tracking Helical Scan Tape Device", invented by Blatchley et al., and having an internal docket number of 9086/108 and a serial number of 09/192,808, filed concurrently herewith on November 16, 1998, and co-pending U.S. patent application entitled "Method And System For Monitoring And Adjusting Tape Position Using Control Data Packets", invented by McAuliffe et al., and having an internal docket number of 9086/107 and a serial number of 09/193,030, filed concurrently herewith on November 16, 1998, and co-pending U.S. patent application entitled "Variable Speed Recording Method and Apparatus for a Magnetic Tape Drive", invented by Beavers et al., and having an internal docket number of 9086/101 and a serial number of 09/176,079, filed on October 20, 1998, and

co-pending U.S. patent application entitled "Overscan Helical Scan Head for Non-Tracking Tap Subsystems Reading at up to 1X Speed and Method for Simulation of Same", invented by Blatchley et al., and having an internal docket number of 9086/104 and a serial number of 09/176,013, filed on October 20, 1998, and co-pending U.S. patent application entitled "Fine Granularity Rewrite Method and Apparatus for Data Storage Device", invented by Zaczek, and having an internal docket number of 9086/106 and a serial number of 09/176,015, filed on October 20, 1998, and co-pending U.S. patent application entitled "Multi-level Error Detection and Correction Technique for Data Storage Recording Device", invented by McAuliffe et al., and having an internal docket number of 9086/102 and a serial number of 09/176,014, filed on October 20, 1998, all of which are commonly owned and all of which are hereby incorporated by reference.

Background of the Invention

Data storage devices, which are used in both short- and long-term capacities, are an integral part of modern computer systems. While factors such as costs, device form factor, storage media size and capacity, and recording and recovery times are of high importance, of primary concern is the ability to maintain data integrity.

Techniques have been developed to detect data errors. One commonly used error detection technique is known as a polynomial or cyclical redundancy check (CRC). CRC generators are preloaded with a generating polynomial G(x) and treat data blocks as a polynomial D(x). The data polynomial D(x) is divided by the generating polynomial G(x) to obtain a quotient Q(x) and remainder R(x), such that $D(x)/G(x) = Q(x) + R(x)$, where

R(x) is known as the cyclic redundancy check (CRC). CRCs are generated over the data and recorded along with the data to the storage media. During the recovery of the data from the storage media during a later session, the CRC is regenerated from the recovered data and compared to the recorded CRC accompanying the data to determine whether or not the data contains an error. CRCs are useful in detection of a high percentage of data errors; however, because binary division is based on performing a logical exclusive OR (XOR) of a series of bits, and therefore the right combination of position and number of errors could result in the same CRC value as that of the correct value of a data packet, CRC checks cannot guarantee accuracy. Accordingly, occasionally a packet containing an error is falsely accepted as a good packet because the error is undetectable by the CRC generator using the particular predefined generating polynomial. This falsely detected "good" packet is herein termed a "rogue" packet.

Accordingly, in order to ensure data integrity, a need exists for a method of detecting and correcting rogue packets in data recovery systems.

Summary of the Invention

The present invention is a novel method for detecting rogue packets in a data buffer. In accordance with the invention, a segment-level CRC is generated across the data and during the recording session by a segment CRC generator and recorded along with the data to the storage medium.

During a data recovery session, only those packets which pass a packet-level error detection test are allowed into the data buffer. When a buffer segment is filled with so-called "good" packets, and the segment-level

CRC associated with the segment is recovered from the storage medium, a segment-level error detection test is performed across the segment-level CRC and the contents of the buffer segment. The segment-level error detection test will detect an error if the segment contains a rogue packet.

If the segment contains a rogue packet, a Reed-Solomon syndrome pair is calculated across each row in the buffer segment. If both syndromes are zero, the row does not contain a rogue packet. If both syndromes are not zero, the syndrome pair is used to calculate the location of a potential rogue packet. Once the location of all the potential rogue packets are determined, the potential rogue packets are effectively erased and error correction is performed on the segment to attempt to correct the rogue packets.

In accordance with one embodiment of the invention, a data segment includes a plurality of data packets arranged in a square array. For each data segment, there are two error correction code (ECC) packets per row, column, and diagonal that are used to correct either one or two missing packets in any data row, column, or diagonal. The ECC packets contain Reed-Solomon two redundancy data. Each data segment has a corresponding packet status table (PST) that contains an entry for each data and ECC packet. Each PST entry indicates whether its corresponding packet has been received and is valid. In addition, the PST contains a count of the number of good packets present in each row, column, and diagonal, and a total number of good data packets in the segment. These counts are used by a segment correction processor (SCP) to determine when correction

can be applied to the segment. The SCP is responsible for all segment correction activities.

Using a Reed-Solomon correction code with two redundant code words provides the ability to locate a single error and correct it, or to correct two errors if the locations are known. To correct two errors, the known locations must be erased or reset to zero so that they do not contribute an error term to the syndrome calculations.

During normal operation, the SCP uses only erasure correction for correcting either single- or two-packet errors. Under error recovery conditions, after a segment CRC test has failed, the SCP determines the location of a rogue packet using the first form of error location and correction. The SCP calculates the syndromes for each row. The resulting syndromes will both be zero if the row has only good packets. If the row has one or more rogue packets, the syndromes are non-zero. The SCP then uses both syndrome terms to calculate the location of a potential rogue packet. If there is only one rogue packet, the corresponding PST entry is marked as not received, and its corresponding row, column, and diagram count, along with the total good packet count are each decremented, and the SCP is restarted for normal correction. The SCP will then correct the missing packet and repeat the segment CRC test. Assuming that it passes the segment CRC test, the segment is released to the host for reading and the read process continues.

The rogue packet detection portion of the SCP searches each row including the row ECC and column ECC. The SCP can determine a possible column location in each row that one or more rogue packets are located in.

A simple range check of the rogue packet column location can give some credence to the result. Namely, if it exceeds the number of columns in the PST, there is probably more than one rogue packet in the row.

After a rogue packet is found, rogue packet detection is resumed until the last row is processed. Each time a rogue packet is located, the SCP sets a "rogue found" bit. Each time a rogue packet is located, or the last row has been processed, the SCP sets a "done" bit.

Brief Description of the Drawing

The invention will be better understood from a reading of the following detailed description taken in conjunction with the drawing in which like reference designators are used to designate like elements, and in which:

FIG. 1 is a block diagram of a system in which the invention operates;

FIG. 2 is a diagram illustrating one embodiment of the contents and organization of a buffer segment implemented in accordance with the invention;

FIG. 3 is a diagram illustrating one embodiment of a packet status table (PST) entry;

FIG. 4 is a diagram illustrating an example embodiment of a PST implemented in accordance with the invention;

FIG. 5 is a flowchart illustrating one embodiment of a method 500 for performing error correction during a normal recovery session; and

FIG. 6 is a flowchart of one embodiment of a method in accordance with the invention for detecting the location of a rogue packet.

Detailed Description

A technique for detecting and correcting rogue packets in a data recording/recovery device in order to provide increased data integrity is described in detail hereinafter. For purposes of illustration, the invention is described in the context of a magnetic tape drive; however, the method of the invention may be employed in any application which performs error detection and correction on data.

Turning now to FIG. 1, there is a shown a block diagram of a system 2 in which the invention operates. System 2 includes a data buffer 10, an associated packet status table (PST) 12, read logic circuitry 4, a data packet error detector 6, a segment control processor (SCP) 8, and a segment CRC generator 14, a segment ECC generator 16 and a segment error correction circuit 18. Read logic circuitry 4 processes incoming data DATA_SIGNAL from a read channel (not shown) that recovers data from a storage media (not shown), such as a magnetic tape. Read logic circuitry 4 decodes and formats the data into packets 5. Each data packet 5 includes a packet-level error detection code such as a packet-level CRC code that is generated over the data portion of the packet. Error detection is performed on each packet 5 by packet error detector 6. If an error is detected in a packet 5, the error is either corrected using error correction techniques or the packet 5 is discarded. A packet is not allowed into the data buffer 10 unless and/or until it passes the packet-level error detection test. If a packet 5 passes the error detection test because no errors are detected in the packet 5, it is sent to its appropriate location in data buffer 10. Various techniques, beyond the scope of the present invention, are known in the art for determining the

appropriate location of a packet 5 in the data buffer 10. In the illustrative embodiment, data is sent to and from a host system (not shown) in data segments 11. Data buffer 10 is large enough to store multiple data segments 11 at any given time.

In accordance with the invention, during a data recording session a segment error detection code such as a cyclical redundancy code, hereinafter "recorded segment CRC", is generated over a given data buffer segment 11 and recorded along with the segment data to the storage medium. Various means for associating a recorded segment CRC with a given segment 11 could be implemented. For example, the recorded segment CRC could be stored in a special control packet that is associated with the segment 11 and identifiable as such by the read logic circuitry 4.

During a data recovery session, read logic circuitry 4 controls the decoding and placement of data packets in buffer 10. Again, no data packet 5 is allowed in data buffer 10 unless it passes the packet-level error detection test without errors.

FIG. 2 illustrates one embodiment of the contents and organization of a segment 11 of a multi-segment data buffer 10 implemented in accordance with the invention. In this embodiment, segment 11 comprises a set of fixed-size buffer packets 15, each for storing one packet 5. In the illustrative embodiment, the segment 11 comprises 1024 buffer packets 15 arranged in a 32-by-32 array.

Segment 11 also comprises a pair of error correction code (ECC) packets 17 P and Q for each row, column, or diagonal that are used to correct either one or two missing packets in its associated row, column, or

diagonal. The row, column, and diagonal ECC packets 17 consist of a pair of Reed-Solomon redundancy terms P and Q.

As known by those skilled in the art, Reed-Solomon error correction techniques allow the number of correctable errors to be designed into a system according to the number of redundant terms. The Reed-Solomon error correction techniques implemented in the illustrative embodiment of the invention are described in detail in "_____", pp. 145-152. In the illustrative embodiment, a two-redundancy code is used by calculating a pair of redundant terms P and Q for each row, column and diagonal. The redundant terms P and Q are solutions to the equations:

$$S_0 = B_{31} \rho B_{30} \rho B_{29} \rho \, J \, \rho B_1 \rho B_0 \rho P \rho Q = 0; \text{ and}$$

$$S_1 = a^{34}B_{31} \rho a^{33}B_{30} \rho a^{32}B_{29} \rho \, J \, \rho a^4B_1 \rho a^3B_0 \rho a^2P \rho a^1Q = 0,$$

where S_0 and S_1 are the syndromes, P and Q are the redundant terms, a^1 through a^{34} are the polynomial coefficients, and B_0 through B_{31} are the bit values. As evident from the above equations, the value of S_0 is simply the parity of all the terms B_{31} through B_0 and P and Q. The syndrome is the same error bit pattern, but it has been raised to a different power of *a* depending on the position of the error symbol in the block. If the position of the error is represented by symbol k, then $S_0 \times a^k = S_1$. Thus, $a^k = S_1 / S_0$. The error symbol can be located by multiplying S_0 by various powers of *a* until the product is the same as S_1. The value of *k* that results in $a^k = 1$ corresponds to the position of the symbol that contains the error. Once the position of the error is known, the correct value of the erroneous term is obtained by adding S_0 to it. The above-described process is used to locate and correct a single error when the location of the error is unknown.

Reed-Solomon two-redundant term systems can also be used to correct two terms if the location of the errors is known. One method of doing this is to set the erroneous terms to zero, (i.e., "erase" the terms) and then calculate the syndromes S_0 and S_1 and solve for the erased terms.

As previously described, as data is decoded from incoming data stream DATA_SIGNAL, read logic circuitry 4 places those data packets 5 that pass the packet-level error detection test in their correct buffer packet 15 locations in segment 11. In addition, when the recorded segment CRC associated with the segment 11 is recovered, it is stored in one of the buffer packets 15, or, in the alternative, is stored temporarily in a recorded segment CRC register, shown at 20 in FIG. 2.

In the illustrative embodiment, each segment 11 has a corresponding PST 12 which contains an entry corresponding to each buffer packet 15 and ECC packet 17 in the segment 11 and which is updated as data packets 5 are recovered and placed in the data buffer 10. Each PST entry 30 indicates whether its associated buffer packet 15 has been received and is valid. In one embodiment, illustrated in FIG. 3, each PST entry comprises a byte with bits as follows: VALID_PACKET (bit 7) which is set if the corresponding packet has been received and contains no errors detectable by the packet-level error detector; LOCKED (bit 6) which is a semaphore lock bit that is set when the PST is being updated; PACKET_CORRECT (bit 5) which is set to indicate that the packet was corrected by packet correction logic; SEGMENT_CORRECT which is set to force the correction of the associated buffer packet 15 by segment-level error correction circuitry 18. FIG. 4 illustrates an example embodiment of a PST 12 implemented in accordance

with the invention. Each PST 12 also contains a good packet count 19 for each row, column, and diagonal which indicates the total number of good packets present in its associated row, column, and diagonal, and a total packet count 32 which indicates the total number of good packets present in the segment 11. The counts are used by the SCP 8 to determine when correction can be applied to the segment 11. Each PST 12 also includes a missing packet pointer for each row, column, and diagonal that temporarily stores a pointer to a missing packet in its associated row, column, or diagonal.

The segment correction processor (SCP) 8 manages all segment correction activities. During a recovery session, SCP 8 monitors the PST 12 to locate as-yet "missing" buffer packets 15 that can be reconstructed through error correction. For example, as a segment 11 begins to fill, if a row, column, or diagonal becomes full less one or two "missing" packets, the segment error correction logic 18, with the assistance of the segment ECC generator 16 and under the control of SCP 8, is able to reconstruct the missing packets. Because every packet lies in a row, column, and diagonal, the reconstruction of one or two missing packets along one of the three different patterns may fill out at least one of the other two patterns enough such that the missing packets located in that pattern can then be reconstructed. For example, if a row is missing only a single packet, error correction is performed on the entire row using the two unique redundancy code row ECC packets P 152 and Q 153 to reconstruct the missing packet. The reconstruction of the missing packet may fill out the diagonal that the reconstructed packet lies in enough that it is only missing a single other

packet. Error correction is then performed on the reconstructed packet's diagonal to reconstruct the missing packet in the diagonal. This process continues until all of the packets in the segment are recovered or reconstructed.

In the preferred embodiment, multi-layer redundancy is provided by pairs P and Q of redundant codes generated over a plurality of patterns of packets 15 of a segment 11 of buffer 10. In the illustrative embodiment, the plurality of patterns include rows, columns, and diagonals. Other patterns may be used. In addition, error correction is performed over all of the first redundant codes P for both row and column ECC codes, and all of the second redundant codes Q for both row and column ECC codes to generate special ECC codes 158.

Segment correction processor (SCP) 8 communicates with a Reed-Solomon syndrome generator 16 and error correction circuit 18 that provides for a one- or two-packet correction in any row, column, or diagonal of segment 11 using erasure pointer correction. Row redundancy is provided by row ECC packets P 152 and Q 153 for each row (0 through 31) of buffer segment 11. Column redundancy is provided by column ECC packets P 154 and Q 155 for each column (0 through 31) of buffer segment 11.

Diagonal redundancy is provided by diagonal ECC packets P 156 and Q 156 for each diagonal (0 through 31) of buffer segment 11. The diagonals are defined to each include 32 packets. In one embodiment, a given diagonal N starts at column 0 and row N, and the next element of the diagonal is found by incrementing both the current column number and

current row number, modulo 32. Table 1 illustrates the calculation of the diagonals for this embodiment.

Table 1

DIAGONAL #	STARTING ROW	STARTING COLUMN	ENDING ROW	ENDING COLUMN
0	0	0	31	31
1	1	0	0	31
2	2	0	1	31
N	N	0	N-1	31

Four special ECC packets 158 are generated from the 64 column ECC packets. These four special ECC packets 158 are used to correct missing row ECC packets 152, 153 or column ECC packets 154, 155.

FIG. 5 is a flowchart illustrating one embodiment of a method 500 for performing error correction during a recovery session as implemented in SCP 8. SCP is triggered in step 502 to begin segment correction. Once the SCP is triggered to begin, in step 504 SCP 8 scans the PST 12 to determine whether the buffer segment contains any missing packets. In the illustrative embodiment, SCP 8 scans the good packet count 19 of each PST entry 30 corresponding to each row, each column, and each diagonal. If all rows, columns, and diagonals have a good packet count 19 equal to the maximum respective number of rows, columns and diagonals (e.g., 32 in the illustrative embodiment), all of the data packets 15 are received and valid, and the normal segment correction method is complete.

If any data packets are missing from the data buffer, in step 506 the SCP 8 scans each instance of every predefined buffer packet pattern to determine whether any one or two packets is correctable. In the illustrative embodiment, SCP scans the good packet count 19 of each PST entry 30 corresponding to each row, each column, and each diagonal. If any row, column, or diagonal has a good packet count 19 that is less by one or two of a total row/column/diagonal count (e.g., 32 in the illustrative embodiment), that particular row/column/diagonal instance is correctable to recover the missing packet(s) of that instance. Step 506 is repeated until a correctable instance is detected or the buffer segment 11 becomes full (indicated in the illustrative embodiment by the total packet count 32 reaching a value of 1220).

If a correctable instance is detected, packet correction is performed in step 508. In the preferred embodiment, packet correction is performed in accordance with Reed-Solomon error correction principles. The SCP 8 uses erasure correction for correcting either single-or two-packet errors. Using a Reed-Solomon correction code with two redundant code words P and Q provides the ability to locate a single error and correct it, or to correct two errors if the locations are known. To correct two errors, the known locations are effectively erased so that they do not contribute an error term to the syndrome S_0, S_1 calculations. In addition, it is possible to correct only one location with erasure correction. Steps 504 - 508 are repeated until all of the data packets 15 and ECC packets 17 are present in the segment.

When all of the buffer packets 15 and ECC packets 17 in a buffer segment 11 are filled in by the read logic circuitry 4, segment-level error

detection is performed to determine whether the segment contains any rogue packets. This is performed under the control of SCP 8 by generating a segment-level CRC over the recorded segment CRC and all of the buffer packets 15 that contain a valid packet 5 and ECC packets 17. In the illustrative embodiment, the CRC generating polynomial used to calculate the segment CRC is:

$$X[32] + X[26] + X[23] + X[22] + X[16] + X[12] + X[11] + X[10] + X[8] + X[7] + X[5] + X[4] + X[2] + X[1] + X[0].$$

If the segment CRC calculated over all buffer packets 15 containing valid data and the recorded segment CRC by segment CRC generator 14 has a value of zero, all of the data in the segment 11 is correct and segment 11 does not contain any rogue packets.

Because the segment-level CRC is calculated over all the data packets in the segment 11, rogue packets that passed the packet-level CRC test, and that are therefore allowed in the buffer 10, generally cause the segment-level CRC test to fail. The detection of the presence of a rogue packet in a segment 11 is heightened by the use a different CRC generating polynomial for the segment CRC calculation.

Upon detection of the presence of a rogue packet in a segment 11 by the failure of the segment-level CRC test, the packet status table (PST) 12 is employed in the detection of the location of rogue packets. SCP 8 determines the location of a rogue packet by calculating the ECC syndromes S_0, S_1 for each row ROW[0:31] of the segment 11. The resulting syndromes S_0, S_1 should both be zero if the row contains only good packets. If the row has one or more rogue packets, the syndromes S_0 and S_1 will be non-zero.

The SCP 8 then uses both syndrome terms S_0, S_1 to calculate the location of a potential rogue packet. The calculated location is only potentially correct using the two-syndrome technique of the illustrative embodiment since only a single error can be corrected using two syndrome terms S_0, S_1 when the location of the error is unknown. Thus, if the row contains more than one rogue packet, the calculation of the location *k* of the erroneous packet using only two syndrome terms S_0, S_1 will be incorrect. However, as will be appreciated by those skilled in the art, the addition of redundant terms and calculation of additional syndromes will allow the detection of the location of additional rogue packets. After the location of a rogue packet has been determined, the packet is effectively erased by modifying the PST entry corresponding to the location of the rogue packet to indicate that the rogue packet has not been received and is not valid. This is done in the illustrative embodiment by clearing the PACKET_VALID bit. The SCP 8 is restarted to correct the rogue packet.

FIG. 6 is a flowchart of one embodiment of a method 600 for detecting the location of a rogue packet. Method 600 is triggered by the detection of a segment CRC failure. In the illustrative embodiment, upon detection of a segment CRC failure, the SCP 8 is signaled to begin executing method 600 for detecting the location of and correcting rogue packets. In a first step 602, SCP 8 causes a pair of syndromes S_0, S_1 to be generated for each row [0:31] of the buffer segment. For each row, SCP 8 detects whether the pair of syndromes S_0, S_1 are both zero in step 604, and if not, causes segment ECC generator 16 to use the syndrome pair S_0, S_1 to locate the column position of a possible rogue packet in step 606. Once the location of a

possible rogue packet is determined, the contents of that location are effectively erased in step 608, and in step 610 the row, column, and diagonal counts associated with the erased location, along with the total good packet count, are decremented, and the missing pointers are reset to zero. If more rows exist to be processed, as determined in step 612, steps 604 through 612 are repeated. When all rows have been processed, the SCP 8 is restarted in step 614. At this point, the SCP 8 detects that there are packets missing from data segment 11, and the missing packets are corrected where possible. Once all the missing packets are filled in, the segment CRC test is re-executed. If all of the possible rogue packet locations were correctly identified, the segment CRC test will pass.

As part of an error recovery procedure, it may be necessary to block specific packets from being read and stored in the data segment. This is possible by setting the SEGMENT_CORRECT bit in the PST entry associated with the packet. The read logic can look at this bit before allowing a packet to be stored in the buffer at the PST entry's associated buffer packet location. This forces these packets to be corrected by the SCP and blocks the same packet from being read from tape on a subsequent reread.

Although the invention has been described in terms of the illustrative embodiments, it will be appreciated by those skilled in the art that various changes and modifications may be made to the illustrative embodiments without departing from the spirit or scope of the invention. It is intended that the scope of the invention not be limited in any way to the illustrative

embodiment shown and described but that the invention be limited only by the claims appended hereto.

WHAT IS CLAIMED IS:

1. A method for detecting a rogue packet in a segment of a data buffer, said data buffer (10) comprising a plurality of packets (15), said plurality of packets (15) mapping into a plurality of instances of a pattern, said method comprising the steps of:

for each of said plurality of instances of said pattern:

generating (602) a plurality of syndromes across said packets in said instance of said pattern;

determining (604) that no rogue packet exists in said instance of said pattern if said syndromes all equal zero; and

calculating (606) a location of a possible rogue packet in said instance of said pattern using said calculated syndromes if said calculated syndromes do not all equal zero.

2. A method in accordance with claim 1, comprising:

erasing (608) said location of said possible rogue packet; and

performing (614) error correction on said erased location.