# GENERATING AND COMMUNICATING SOURCE IDENTIFICATION INFORMATION TO ENABLE RELIABLE COMMUNICATIONS

## PRIORITY

**[0001]** This application claims the benefit under 35 U.S.C. 119(e) of and is a non-provisional of U.S. Application Serial No. 60/971,884 filed September 12, 2007 and U.S. Application Serial No. 61/093,277 filed August 29, 2008. All of these applications are incorporated by reference in their entirety for all purposes.

## CROSS-REFERENCES

**[0002]** The following references are included here and are incorporated by reference for all purposes:

**[0003]** U.S. Patent No. 6,307,487 entitled "Information Additive Code Generator and Decoder for Communication Systems" issued to Luby (hereinafter "Luby I");

**[0004]** U.S. Patent No. 7,418,651 entitled "File Download and Streaming System" to Luby, et al. (hereinafter "Luby II");

**[0005]** U.S. Patent No. 7,068,729 entitled "Multi-Stage Code Generator and Decoder for Communication Systems" issued to Shokrollahi, et al. (hereinafter "Shokrollahi I");

**[0006]** U.S. Published Patent Application No. 2006/0036930 published February 16, 2006 and entitled "Method and Apparatus for Fast Encoding of Data Symbols According to Half-Weight Codes" to Luby, et al. (hereinafter "Luby III"); and

**[0007]** U.S. Patent No. 6,856,263 entitled "Systems and Processes for Decoding Chain Reaction Codes Through Inactivation" issued to Shokrollahi, et al. (hereinafter "Shokrollahi II").

**[0008]** U.S. Published Patent Application No. 2007/0300127 published December 27, 2007 entitled "Code Generator and Decoder for Communications Systems Operating Using Hybrid Codes to Allow for Multiple Efficient Users of the Communications Systems" to Watson et al. (hereinafter "Watson").

**[0009]** U.S. Patent U.S. Patent No. 7,249,291, issued July 24, 2007 entitled "System and Method for Reliably Communicating the Content of a Live Data Stream" issued to Jens Rasmussen, et al. (hereinafter "Rasmussen").

## FIELD OF THE INVENTION

**[0010]** The present invention relates to encoding and decoding data in communications systems, and more specifically to communication systems that encode and decode data to account for errors and gaps in communications.

## BACKGROUND OF THE INVENTION

**[0011]** Transmission of files and streams between a sender and a recipient over a communications channel has been the subject of much literature. Preferably, a recipient desires to receive an exact copy of data transmitted over a channel by a sender with some level of certainty. Where the channel does not have perfect fidelity, which characterizes most physically realizable systems, one concern is how to deal with data that is lost or corrupted in transmission. Lost data (erasures) are often easier to deal with than corrupted data (errors) because the recipient cannot always recognize when the transmitted data has been corrupted.

**[0012]** Many error-correcting codes have been developed to correct erasures and/or errors. Typically, the particular code used is chosen based on some information about the infidelities of the channel through which the data is being transmitted, and the nature of the data being transmitted. For example, where the channel is known to have long periods of infidelity, a burst error code might be best suited for that application. Where only short, infrequent errors are expected, a simple parity code might be best.

**[0013]** Another commonly used technique for providing reliable data delivery is the use of data retransmission. For example, the well-known TCP/IP protocol uses packet retransmission of lost or missing packets to ensure reliable delivery of data. Another example is the HTTP protocol, which is built on top of TCP/IP and uses the reliability of the TCP/IP protocol to provide reliable data delivery. Enhancements of other protocols to use retransmission, such as the RTP protocol, have also been suggested as a way of coping with lost or missing packets at receivers.

**[0014]** Another technique that has been suggested for the improvement of streaming applications is to send initial data for a stream in one channel to a receiver and then to transition over to sending the main data stream in a second channel to a receiver. For example, Rasmussen suggests such a method. As another example, initial data may be sent to a receiver via a unicast connection to ensure that the receiver has enough data quickly to start the play-out of a video or multi-media stream, and then the receiver may switch over to a multicast connection to receive further data for the stream.

**[0015]** "Communication", as used herein, refers to data transmission, through space and/or time, such as data transmitted from one location to another or data stored at one time and used at another. The channel is that which separates the sender and receiver. Channels in space can be wires, networks, fibers, wireless media, etc. between a sender and receiver. Channels in time can be data storage devices. In realizable channels, there is often a nonzero chance that the data sent or stored by the sender is different when it is received or read by the recipient and those differences might be due to errors introduced in the channel.

**[0016]** Data transmission is straightforward when a transmitter and a receiver have all of the computing power and electrical power needed for communications, and the channel between the transmitter and receiver is reliable enough to allow for relatively error-free communications. Data transmission becomes more difficult when the channel is in an adverse environment, or the transmitter and/or receiver has limited capability. In certain applications, uninterrupted error-free communication is required over long periods of time. For example, in digital television systems it is expected that transmissions will be received error-free for periods of many hours at a time. In these cases, the problem of data transmission is difficult even in conditions of relatively low levels of errors.

**[0017]** Another scenario in which data communication is difficult is where a single transmission is directed to multiple receivers that may experience widely different data loss conditions. Furthermore, the conditions experienced by one given receiver may vary widely or may be relatively constant over time.

**[0018]** One solution to dealing with data loss (errors and/or erasures) is the use of forward error correcting (FEC) techniques, wherein data is coded at the transmitter in such a way that a receiver can correct transmission erasures and errors. Where feasible, a reverse channel from the receiver to the transmitter enables the receiver to relay information about these errors to the transmitter, which can then adjust its transmission process accordingly. Often, however, a reverse channel is not available or feasible, or is available only with limited capacity. For example, in cases in which the transmitter is transmitting to a large number of receivers, the transmitter might not be able to maintain reverse channels from all the receivers. In another example, the communication channel may be a storage medium.

**[0019]** For example, data may be transmitted chronologically forward through time, and causality precludes a reverse channel that can fix errors before they happen. As a result, communication protocols often need to be designed without a reverse channel or with a

limited capacity reverse channel and, as such, the transmitter may have to deal with widely varying channel conditions without prior knowledge of those channel conditions. One example is a broadcast or multicast channel, where reverse communication is not provided, or if provided is very limited or expensive. Another example where such a situation is relevant is a storage application, where the data is stored encoded using FEC, and then at a later point of time the data is recovered, possibly using FEC decoding.

**[0020]** Another solution is based on retransmission that is based on a receiver understanding which packets are not received and then sending requests to the sender to retransmit those missing packets. The identification of which packets are missing is often based on sequence numbers carried within the packets. Examples of such protocols include TCP/IP, NORM, RTP with retransmission, etc.

**[0021]** Another solution is based on the combination of FEC and retransmission. In this case, FEC may be proactively sent and then for example only if the receiver loses more than can be recovered by the FEC decoder does the receiver request retransmission of packets, or transmission of additional FEC packets in order to provide enough packets to the FEC decoder for recovering the original source packets. As another example, no FEC may be sent initially, and only if there are missing packets would the receiver request additional packets that may be FEC packets that can be used to recover the original source packets. For example, this may be a solution of interest in the case of sending the original source stream via multicast and then the requested packets are also sent in either in the same stream or in a different multicast stream. For example, different receivers may lose different numbers of packets, and then a sender sending the requested packets may send for example the maximum number of FEC packets requested by all receivers that will allow all receivers to recover the original source independent of their individual packet loss patterns.

**[0022]** In the case of a packet protocol used for data transport over a channel that can lose packets, a file, stream, or other block of data to be transmitted over a packet network is partitioned into source symbols (that may all be of equal size or that may vary in size depending on the block size or on other factors). Encoding symbols are generated from the source symbols using an FEC code, and the encoding symbols are placed and sent in packets. The "size" of a symbol can be measured in bits, whether or not the symbol is actually broken into a bit stream, where a symbol has a size of M bits when the symbol is selected from an

alphabet of $2^M$ symbols. In such a packet-based communication system, a packet-oriented erasure FEC coding scheme might be suitable.

**[0023]** A file transmission is called reliable if it enables the intended recipient to recover an exact copy of the original file despite erasures and/or other corruption of the data transmitted over a network. . A stream transmission is called reliable if it enables the intended recipient to recover an exact copy of each part of the stream in a timely manner despite erasures and/or corruption within the network. Both file transmission and stream transmission can instead be not entirely reliable, but somewhat reliable, in the sense that some parts of the file or stream are not recoverable or, for streaming, some parts of the stream might be recoverable but not in a timely fashion. It is often the goal to provide as high reliability as possible depending on some constraining conditions, where examples of constraints might be timely delivery for streaming applications, or the type of network conditions over which a solution is expected to operate.

**[0024]** Packet loss often occurs because sporadic congestion causes the buffering mechanism in a router to reach its capacity, forcing it to drop incoming packets. Other causes of packet loss include weak signal, intermittent signal, and noise interference wherein corrupted packets are discarded. Protection against erasures during transport has been the subject of much study.

**[0025]** In a system in which a single transmission is directed to more than one receiver, and in which different receivers experience widely different conditions, transmissions are often configured for the some set of conditions between the transmitter and any receiver, and any receivers that are in worse conditions may not receive the transmission reliably.

**[0026]** Erasure codes are known which provide excellent recovery of lost packets in such scenarios. For example, Reed-Solomon codes are well known and can be adapted to this purpose. However, a known disadvantage of Reed-Solomon codes is their relatively high computational complexity. Chain reaction codes, including LT™ chain reaction codes and Raptor™ multi-stage chain reaction ("MSCR") codes, provide excellent recovery of lost packets, and are highly adaptable to varying channel conditions. See, for example, Luby I, which describes aspects of chain reaction codes, and Shokrollahi I, which describes aspects of multi-stage chain reaction codes. Herein, the term "chain reaction code" should be understood to include chain reaction codes or multi-stage chain reaction codes, unless otherwise indicated.

**[0027]** Retransmission protocols are also known to be a good way to recover lost packets. TCP/IP, NORM, UDP and RTP based retransmission protocols are all examples of such retransmission protocols. In addition, using a combination of erasure code protocols and retransmission protocols can be quite useful for recovering lost packets. Retransmission protocols include any protocol where a receiver request specific packets or numbers of packets to be sent to the receiver, and where a sender may send specific packets or numbers of packets to that receiver, or to groups of receivers, in response.

**[0028]** Some communication systems use transport protocols, such as RTP, that include information usable to identify the position or sequence of a source packet relative to other source packets in the same stream. By providing sequence information for each packet, a receiver may detect and correct packets that are received out of network order. A receiver can also detect when packets have been lost, and when combined with application layer FEC techniques, such as those of DVB-IPI (see, for example, the descriptions in Watson), the receiver may be able to recover the lost packets and effectively mask network reliability imperfections. DVB details are known from the reference for the DVB-IPI standard: "DVB BlueBook A086r4 (03/07)/ETSI Technical Specification 102 034 v1.3.1", which is available at the URL: http://www.dvb.org/technology/standards/a086r4.dTS102034.V1.3.1.pdf.

**[0029]** Many deployed communication systems use transport level protocols that do not include any form of timing or sequence information. For example, it is common practice in IPTV networks to deliver MPEG2 Transport Stream packets over "raw UDP". The only sequence information available to a receiver is embedded in the audio and video elementary stream, which may be hard to access, or unreliable, and not generally available at the transport level. Consequently, there is no inherent mechanism at the transport level with raw UDP streams that allow a receiver to recognize when a packet is received out of network order or to identify missing packets. While well-known mechanisms, such as application layer FEC ("AL-FEC"), may be used to efficiently recover lost packets, the absence of transport level sequence information on source stream packets limits the direct applicability of such recovery techniques. These same issues apply to retransmission solutions and to combinations of retransmission and AL-FEC solutions. Thus, this is a general problem with some existing approaches.

**[0030]** Another problem with some communications systems is that the parts are interrelated. In some cases, it may be necessary or desirable to increase the reliability of a

communications system after deployment. However, while an improvement in network reliability may be needed, it is typically not feasible to replace or upgrade all receiving devices in the network at once or at all. For example, it might turn out that actual network packet loss is higher than initially planned, due to degradations in network reliability, increased traffic load, expansions and/or changes in the network, etc., or the quality of service requirements may need to increase to match competitive services, but it might be impractical to get new receivers out to all nodes of the communications system at once or to distribute them over time and have some receiving stations out of commission until the new receivers arrive.

**[0031]** In order to ensure that legacy devices are unaffected by protocol enhancements, such as AL-FEC or retransmission or combinations of AL-FEC and retransmission, used by upgraded or new receivers, it is necessary to continue to deliver source packets using the same transport level protocol. Furthermore, to ensure that source packets delivered to a legacy device do not become more susceptible to burst loss, it is necessary to maintain the same source packet timing distribution or inter-packet timing. In some communication systems, source packets within a source block may be transmitted with a smaller inter-packet gap to allow repair packets to be delivered immediately after the associated source packets; such techniques would increase the exposure of the source stream to burst loss and therefore degrade the transport effectiveness for a legacy device.

**[0032]** In order to deliver the best possible service at the lowest cost, communications systems must simultaneously balance conflicting resource constraints. Network bandwidth is a critical resource constraint. Transmitting and receiving devices need to enable efficient use of network bandwidth in supporting a reliable service. The available CPU processing on receiving devices is typically a severe limitation, meaning that any transport reliability enhancement method must require only a modest amount of computing effort. In addition, it is also often necessary, particularly with streaming media, to limit the incremental latency associated with reliable transport methods so that the end-user does not perceive a reduction in system responsiveness.

## BRIEF SUMMARY OF THE INVENTION

**[0033]** According to one embodiment of the invention, a method of generating Source Identification information or data, referred to as Source Identification data or information, from a source stream is described. The method can operate on an original source stream and

generate relatively compact Source Identification information or data that may be used by a receiving device to identify the position of source packets relative to other source packets, and to identify lost or corrupted source packets. Source Identification information or data generation methods can result in a compact representation of the source packet identification, which enables the Source Identification information for a block of source packets to be delivered using a relatively small amount of network bandwidth. The computational load of deriving the Source Identification information is small.

**[0034]** The Source Identification data may be combined with forward error correction ("FEC") techniques to enable recovery of lost or corrupted source packets. The Source Identification data may also be combined with retransmission techniques, and combinations of FEC techniques and retransmission techniques to enable recover of lost or corrupted source packets. The Source Identification data may also be used with techniques that identify at a receiver in which order a sequence of source packets are sent from a sender to that receiver, even when some sent packets are lost, corrupted or re-ordered.

**[0035]** In another embodiment of the invention, method and apparatus might be provided that enable reliable delivery of Source Identification data so that the identity or relative position of source packets can be determined at a receiving device and by extension, the absence of a received source packet can be derived. To provide reliable delivery of the Source Identification, packets that carry the Source Identification can be transmitted in a manner than minimizes exposure to burst loss or periodic loss patterns. Since the Source Identification data is derived from the source packet content, two or more source packets may share the same the Source Identification data and variations of these methods can protect against such Source Identification collisions at the receiver.

**[0036]** According to another embodiment of the invention, a method of transmitting data over a communications channel is provided, the method comprising the step of transmitting an original unmodified source stream, and transmitting FEC repair data and Source Identification data as additional streams that can be used to efficiently recover lost or corrupted packets in the original unmodified source stream.

**[0037]** According to another embodiment of the invention, a method of transmitting data over a communication channel is provided, the method comprising the step of transmitting an original unmodified source stream, and transmitting Source Identification data as additional streams that can be used to identify which data is missing or corrupted from the received data

and request retransmission. In a further enhancement of this embodiment, retransmitted data is also identified using Source Identification information. In a further enhancement of this embodiment, out of order data reception is identified and re-ordered using Source Identification information. In a further enhancement, Source Identification information is used to both recover lost or corrupted data in the original unmodified source stream using FEC and used to identify which data is missing or corrupted from the original source stream and to request retransmission of missing or corrupted data, or transmission of additional repair data that can be used to recover the data in the unmodified source stream. In a further enhancement of this embodiment, a first repair stream is transmitted along with the original unmodified source stream. In a further enhancement of this embodiment, a second repair stream is transmitted in response to any requests for retransmission. In a further enhancement of this embodiment, one or more of the various streams are cached.

**[0038]** According to another embodiment of the invention, the Source Identification information identifies which block of data within an unmodified source stream of data that source packets are associated with. In this case, a unique identifier for each unmodified source packet may not be able to be generated, i.e., the Source Identification information may be used to identify the block of data a source packet is associated with, but may not be able to differentiate the source packets within a block from one another.

**[0039]** According to another embodiment of the invention, a method of receiving data transmitted over a communications channel is provided, the method comprising the steps of: (1) for a non-FEC enabled receiver or for zero or more of the FEC enabled receivers, receiving the original unmodified source stream; (2) for one or more of the FEC enabled receivers, receiving an original unmodified source stream and some or all of the additional streams of FEC repair data and Source Identification data and using this to efficiently recover lost packets from the original source stream.

**[0040]** According to another embodiment of the invention, a method of receiving data transmitted over a communications channel is provided, the method comprising the steps of: (1) for a non-retransmission enabled receiver or for zero or more of the retransmission enabled receivers, receiving the original unmodified source stream; (2) for one or more of the retransmission enabled receivers, receiving an original unmodified source stream and some or all of the additional streams of retransmission repair data and Source Identification data and using this to efficiently recover lost packets from the original source stream.

**[0041]** According to another embodiment of the invention, a method of receiving data transmitted over a communications channel is provided, the method comprising the steps of: (1) for a non-FEC and non-retransmission enabled receiver or for zero or more of the FEC and retransmission enabled receivers, receiving the original unmodified source stream; (2) for one or more of the FEC and retransmission enabled receivers, receiving an original unmodified source stream and some or all of the additional streams of FEC and retransmission repair data and Source Identification data and using this to efficiently recover lost packets from the original source stream.

**[0042]** According to another embodiments of the invention, a method of receiving data transmitted over a communications channel is provided, the method comprising the steps of: (1) for an FEC enabled or retransmission enabled receiver, determining if a particular stream that is being received has associated FEC repair data or retransmission data available; (2) if FEC repair data or retransmission data is available, receiving some or all of the additional streams of FEC and retransmission repair data as needed and Source Identification data and using this to efficiently recover lost packets from the original source stream.

**[0043]** According to another embodiment of the invention, a method of transmitting data over a communications channel is provided, the method comprising the step of transmitting an original unmodified source stream, and transmitting additional streams of related data that can be used to efficiently recover lost packets in the original unmodified source stream.

**[0044]** As will be clear to those of skill in the art upon review of this disclosure, the methods described herein can be naturally extended to many variations of the above, including sending multiple layers of FEC repair data, sending the source stream to one IP address and the additional streams to other IP addresses, sending retransmission data individually to receivers, sending the same retransmission data via broadcast or multicast to multiple receivers, sending a combination of FEC repair data and retransmission data to receivers, sending FEC repair data to one set of receivers and sending retransmission to a second set of receivers, sending all streams to the same IP address but distinguished by using different port numbers within the packets, sending some of the streams to only certain portions of the receivers and not others, etc.

**[0045]** A better understanding of the nature and the advantages of the embodiments disclosed herein may be realized by reference to the remaining portions of the specification.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0046]** Fig. 1 is a block diagram of a communications system that can generate and use one or more of the Source Identification techniques described herein used in conjunction with FEC repair techniques.

**[0047]** Fig. 2 is a block diagram of a communication system that supports both legacy set-top boxes and FEC-enabled set-top boxes.

**[0048]** Fig. 3 is a state diagram of a FEC-enabled receiver determining whether or not repair is available for a particular stream.

**[0049]** Fig. 4 is a flow chart for a possible FEC-enabled receiver.

**[0050]** Fig. 5A is a flow chart that shows an example of one of the Source Identification data methods.

**[0051]** Fig. 5B is a block diagram that shows an example of one of the Source Identification data methods.

**[0052]** Fig. 6A is a block diagram that shows an example of the formation of some Source Identification data.

**[0053]** Fig. 6B is a flow chart that shows how a receiver may use Source Identification data to calculate identification information for unmodified received source packets.

**[0054]** Fig. 7 is a block diagram that shows an example of the formation of some Source Identification data.

**[0055]** Fig. 8A is a block diagram that shows an example of some of the further details of the formation of Source Identification data.

**[0056]** Fig. 8B is a block diagram that shows an example of a possible Source Identification data format.

**[0057]** Fig. 9 is a block diagram that shows an example of source block boundaries and overlap with neighboring source blocks in the formation of Source Identification data.

**[0058]** Fig. 10 is a block diagram that shows an example of another possible Source Identification data format.

**[0059]** Fig. 11A is a block diagram that shows an example of two identical packets hashing to the same value.

**[0060]** Fig. 11B is a block diagram that shows a sequence-to-position mapping that can handle identical packets.

**[0061]** Fig. 12 is a block diagram that shows an example where non-identical packets hash to the same value.

**[0062]** Fig. 13 is a block diagram that shows how Source Identification data may be FEC protected.

**[0063]** Fig. 14 is a block diagram that shows how FEC repair symbols may be generated from unmodified source packets.

**[0064]** Fig. 15 is a block diagram that shows a possible format for packets that comprise both FEC repair data and Source Identification data.

**[0065]** Fig. 16 is a block diagram that shows a possible partial reconstruction of Source Identification information from Source Identification data carried in a packet.

**[0066]** Fig. 17 is a flow chart that shows an example of how a receiver may use Source Identification information to process received source data.

**[0067]** Fig. 18 is a block diagram that shows lost packets during a reception time-out at a receiver that spans two source blocks.

**[0068]** Fig. 19 is a block diagram that shows lost packets during a reception time-out at a receiver that occurs early in a source block.

**[0069]** Fig. 20 is a block diagram that shows delayed packets within a source block during a reception at a receiver.

**[0070]** Fig. 21 is a block diagram that shows delayed packets between source blocks during a reception at a receiver.

**[0071]** Fig. 22 is a block diagram that shows delayed packets during a reception at a receiver that cross beyond an extended block.

**[0072]** Fig. 23 is a block diagram that shows duplicate packet reception at a receiver within an extended block, with examples in Fig. 23A and Fig. 23B.

**[0073]** Fig. 24 is a block diagram that shows duplicate packet reception at a receiver beyond an extended block.

**[0074]** Fig. 25 is a table that shows some example results of protection of streams with FEC repair data and Source Identification data.

**[0075]** Fig. 26 is a block diagram that shows an example of protecting a stream with FEC repair data and Source Identification data and retransmission repair data.

**[0076]** A further understanding of the nature and the advantages of the inventions disclosed herein may be realized by reference to the remaining portions of the specification and the attached drawings.

## DETAILED DESCRIPTION OF SPECIFIC EMBODIMENTS

**[0077]** It is to be understood that the various functional blocks described herein may be implemented by a combination of hardware and/or software, and that in specific implementations some or all of the functionality of some of the blocks may be combined. Similarly, it is also to be understood that the various methods described herein may be implemented by a combination of hardware and/or software. Thus, where a computational step is performed, which might be described as "we then do step X", it should be understood that such descriptions include electronic hardware and/or software, or the like, performing those steps, typically as part of a communications process and not involving human or manual interaction.

**[0078]** In some embodiments described herein, data to be encoded is segmented into "source blocks," each block comprising a number of packets of data, known as "source packets," with the number of source packets in a block possibly varying between blocks. For each source block, a number of "FEC repair" packets are generated by the encoder, the number also possibly varying between blocks. "Source packets" and "FEC repair", (or "repair") packets have the characteristic that a receiver receiving a combination of source and repair packets greater than or equal in number to the number of source packets has some non-zero probability of recovering all the source packets.

### Source Identification data types and uses

**[0079]** In some embodiments herein, Source Identification data can refer to data from which the positions of source symbols (which can be sent in packets or other units of transport, hereafter referred to more generally as source data units) relative to other source symbols, as well as the positions of lost source symbols, can be identified at a receiving device. As used herein, a Source Symbol Map or Source System Map (both can be referenced as an "SSM") is one form of Source Identification data or information. For the

purposes of this disclosure, a Source Symbol Map and Source System Map are used interchangeably with Source Identification data. In some embodiments herein, Source Identification data can refer to data from which the positions of sent packets in a stream of packets relative to other packets, comprising the positions of both received and lost packets, can be identified at a receiving device. In some embodiments herein, Source Identification data can refer to data that is used at a sender to identify packets or symbols or other units of data that are requested for retransmission from a receiver. Source Identification data can be used at a receiver to recover lost symbols or packets or other units of data from a combination of source and repair symbols or packets. Source Identification data can be used to identify which symbols or packets or other units of data is lost in order to request their retransmission. Source Identification data can also be used to determine the original sending order of packets or symbols relative to other packets or symbols at a receiver. Source Identification data may be considered as a single part, may be partitioned into sub-parts, or may be thought of as a stream, whereas the single part, sub-parts or stream of Source Identification data may be used to identify all or parts of the source data units, for example a part of Source Identification data may be used to identify those source data units within a source block. As one skilled in the art will recognize, there are many other possible uses of Source Identification data that are not enumerated herein, but which can be easily surmised using the methods and processes disclosed herein. Possible different types and uses of Source Identification data include, but are not limited to, the following:

(1) **Determining identification labels for known source data units method** – the ability to determine a short identification label for each known source data unit from the Source Identification data. One example use is that a receiver can send labels for received source data units to a sender so that source data units associated with labels not received can be transmitted from the sender to the receiver. A variant of this method allows identification labels to be determined for a restricted set of possible source data units, for example those source data units that belong to a particular source block, or those source data units that are sent within a specified period of time.

(2) **Determining identification labels for all source data units method** – the ability to determine a short identification label for all source data units, whether the source data unit is known or unknown, from the Source Identification data. One example use is that a receiver can send labels of missing source data units

to a sender so that source data units associated with labels received can be transmitted from the sender to the receiver. A variant of this method allows identification labels to be determined for a restricted set of possible source data units, for example those source data units that belong to a particular source block, or those source data units that are sent within a specified period of time.

(3) **Determining positioning method** – the ability to determine a position for each source data unit with respect to an ordering of all of the source data units, for both known and unknown source data units, where the ordering is determined by the Source Identification data but may be different than the sending order of the data units or different than the order of the source data units within the stream or file. One example use is that a sender and receiver can use this information to determine where to place source data units within a source block for FEC encoding and FEC decoding. A variant of this method allows positions to be determined for a restricted set of possible source data units, for example those source data units that belong to a particular source block, or those source data units that are sent within a specified period of time.

(4) **Determining sequencing method** – the ability to determine the position of each source data unit with respect to a sending order sequence or other source-determined order sequence (such as the order of the source data units in a file, or the order of the data in an audio-video stream) of all of the source data units, for both known and unknown source data units, from the Source Identification data. One example use is that a receiver can use this ordering to determine in what order to deliver data units to an application, for example in what order to write the data units to a file, or in what order to send the data units for a media stream to a media player. A variant of this method allows positions to be determined for a restricted set of possible source data units, for example those source data units that belong to a particular source block, or those source data units that are sent within a specified period of time.

**[0080]** When the determining positioning method is used in conjunction with an FEC method, the ordering of the source symbols within a source block can be according to the positions from the determining positioning method, and in this case the Source Identification method may interact with how the FEC encoding and decoding methods operate. On the

other hand, when the determining sequencing method is used in conjunction with an FEC method, the ordering of the source symbols within a source block can be according to the sequencing, and since the sequencing is generally not dictated by the determining sequencing method, in this case the Source Identification method may be largely independent of how the FEC encoding and decoding methods operate. In some cases, a Source Identification method may be used which is a combination of a determining positioning method and a determining sequencing method, e.g., there may be some initial Source Identification data which is used to determine positions of source symbols within a source block, then FEC decoding is performed to recover source symbols according to their positions, and then there may be some additional Source Identification data that is part of the source block that is used to determine the sequencing of the recovered source symbols for delivery to an application.

**[0081]** When the determining positioning method is used in conjunction with a Retransmission method, source packets that are requested for retransmission by a receiver can be identified by their positions, e.g., the receiver can send the position of a source packet requested for retransmission to a sender. In these cases, since the sequencing order may be different than the position order, the sender may send the retransmitted packets according to their sequence order, and the receiver may determine the sequencing based on the original order of reception of source packets not lost and the order of reception of retransmitted source packets. When the determining sequencing method is used in conjunction with a retransmission method, source packets that are requested for retransmission by a receiver can be identified by their sequencing, e.g., the receiver can send the sequence number of a source packet requested for retransmission to a sender. In some cases, a Source Identification method may be used which is a combination of a determining positioning method and a determining sequencing method, e.g., there may be some initial Source Identification data which is used to determine positions within a source block that is sent during the original transmission of source packets, and then there may be some additional Source Identification data that is sent during retransmission that is used to determine the sequencing of the source packets for delivery to an application, for example the sequence numbers of the retransmitted source packets.

**[0082]** As one skilled in the art will recognize, there are many variants of the above. For example, the Source Identification methods can be used in conjunction with combinations of FEC methods and Retransmission methods, in which case the requests for retransmission

may be for particular source packets, for particular repair packets, or for a specified number of repair packets instead of for particular repair packets.

Example of using a Source Identification Method in conjunction with an FEC Method

**[0083]** Fig. 1 is a block diagram of a communications system that can generate and use the Source Identification techniques described herein used in conjunction with FEC repair methods.

**[0084]** In communications system 100, input stream or file 110 is provided to a Source Identification generator 120. Source Identification generator 120 generates Source Identification data that contains information about the relative positions of source packets or symbols within the stream of packets or symbols. All source symbols or packets may have the same size, which is typically determined by the use of communications system 100, or the symbol size or packet size can vary from use to use or can be varied within a stream. The generations of Source Identification data may be independent of source symbol size or packet size.

**[0085]** The input stream or file 110 may be provided to the FEC encoder 130. In some embodiments, the FEC encoder 130 performs systematic FEC encoding, which generates FEC repair symbols based on one or more of the input source symbols and their positions, and often places these FEC repair symbols in repair packets for transmission.

**[0086]** It is worth noting that there may be source packets constructed from input stream or file 110 that are used for transmission over communication channels, the construction of these source packets may be independent of the source symbol size. There may or may not be a one-to-one mapping between source packets and source symbols. The source packets may or may not be of variable length.

**[0087]** The source packets from input stream or file 110, the Source Identification data from Source Identification generator 120, and the FEC repair symbols from FEC encoder 130, are provided to a transmit module 140. The transmit module 140 may send Source Identification data and FEC repair symbols in packets to a channel 145; the Source Identification data and FEC repair symbols may or may not be combined in packets during this process. These packets, along with the source packets, are transmitted over a channel 145.

**[0088]** The channel 145 may be an erasure channel, but that is not a requirement for proper operations of communications system 100. Typically the original source packets are sent as a logically separate data stream from other packets. Examples of logically separate data streams include, but not limit to, streams sent over different multicast groups, or streams sent to different ports. The data streams sent to channel 145 may be received by receive modules 150 and 155. Receive module 150 is intended to work with FEC decoder, whereas receive module 155 is a legacy device that may not include FEC decoding. It should be noted that the communications system 100 may be designed such that the legacy receive module 155 only receives source packets, or receives a combination of packets but filters out packets other than source packets.

**[0089]** The legacy receive module 155 can be enabled to recognize and handle the source packets from channel 145. If any repair packets or Source Identification data arrive, they can be on a logically separate stream and can be ignored (silently dropped) by the legacy receive module 155. The received stream 185 is produced from legacy receive module 155.

**[0090]** The receive module 150 can logically differentiate Source Identification data, repair symbols from source packets. The source packets and Source Identification data are provided to Source Identification interpreter 160, from which source symbol IDs identifying source symbols carried in source packets can be identified. The Source Identification data, which provides relationship between received source symbol values and their positions, are sent to the FEC decoder 170.

**[0091]** In addition, the FEC decoder 170 also takes as input the FEC repair symbols and the source packets, both are provided from receive module 150. The FEC decoder 170 tries to recover missing source packets, if any. The received source packets, along with any recovered missing source packets from FEC decoder 170, are reassembled if necessary, to produce recovered stream or file 180. The source packets sent as the recovered input stream or file 180, may be sent in exactly or approximately the original order in which they were sent from input stream 110.

**[0092]** There are many variants of the communication system 100. For example, an FEC encoder 130 may not be used, or the FEC encoder 130 may be replaced with a Retransmitter sender module that receives request for retransmission of lost or missing packets and retransmits those packets based on Source Identification data. As another example, an FEC decoder 170 may not be used, or the FEC decoder 170 may be replaced with a Retransmitter

receiver module that determines which packets or symbols are lost or missing based on Source Identification data and sends requests for those missing packets or symbols, and upon receipt of resent packets or symbols places them into their proper order and passes them on as a Recovered stream 180. As another example, FEC encoder 130 and FEC decoder 170 may not be used, and instead a Packet re-orderer may be used in place of an FEC decoder 170 to reorder mis-ordered received source packets based on Source Identification information. As another example, FEC encoder 130 and FEC decoder 170 may be enhanced to comprise handling of both FEC repair and retransmission logic and methods. As another example, some parts of Source Identification data may be sent directly to a receiver, whereas other parts of Source Identification data may be recovered as part of the FEC recovery process.

**[0093]** The communication system 100 can be used for delivery of streams, files, or other types of data.

Seamless Upgrade Paths

**[0094]** There are a variety of reasons why a full upgrade of a communications system may take a prolonged period of time. For example, it could be the case that older devices may not have the capability to support the upgraded system due to limitations in computational resources or memory. As another example, some devices are designed with the characteristic that they cannot be upgraded, and thus the upgrade can only occur when the device is replaced with a new device in which the upgrade is already installed. In other cases, even if in theory an upgrade is possible, it may in practice be too expensive or risky to upgrade a large number of devices because of the expense of installing upgrade software remotely on millions of devices and making sure that the upgrade is working correctly, as well as troubleshooting problems during installation, etc. Furthermore, if there are a large number of devices installed and they all need to be upgraded before the system is operational again, then it could be practically impossible to shut down the system during the upgrade, as would be the case with a large operational deployment of an IPTV service with thousands, tens of thousands, or even millions of receivers, which could take days or months to fully upgrade.

**[0095]** In such cases, a seamless upgrade path is highly desirable but is often difficult to accomplish.

**[0096]** With respect to deployment of FEC protection of streaming data within a system that did not previously use FEC, there is a well-known potential seamless upgrade path. The way this is done is to design the application of the FEC in such a way that the original source

stream is sent, and then additional FEC repair data is sent in a logically separate stream that can be used by upgraded receivers to support a much higher quality stream playout, while at the same time older receivers that have not been upgraded can simply ignore the FEC repair data and still operate as before using the original source stream of data. The possibility of this seamless upgrade path is one of the primary reasons that systematic FEC codes are preferable for streaming applications. A systematic FEC code has the property that the original source data is included and sent as part of the FEC encoded data, and in this context the additional FEC data that is generated and sent is called the FEC repair data.

**[0097]** An obstacle to this seamless upgrade path is that the association between the FEC repair data and the original source data stream from which the FEC repair data is generated needs to be determined at receivers. In particular, for many FEC schemes, source symbols or packets that comprise an original source data stream can be identified and/or sequenced by source symbol identifiers or packet identifiers, and source symbol identifiers or packet identifiers can be used by an FEC decoder to recover missing or lost source data from received FEC repair data. However, in many streaming systems, such source symbol or packet identifiers are not included, are only partially included, are not completely reliable, or are difficult to access in the original source stream. While there are transport protocols, such as RTP, that natively include sequence information at each source packet which can be used as a source packet identifier and/or sequencer, many deployed communication systems do not make use of these protocols, as a result no timing or sequence information is included in the original source packets or accessible at the transport layer. As an example, IPTV deployments often use UDP packets that carry MPEG2 TS units, and in these deployments identifiers are often not readily accessible or available at the transport level. Furthermore, adding identification data to symbols or packets in the original source stream is not possible for a variety of reasons. Among these reasons are that existing deployed devices would not work if the original source stream is modified in any way, or that the existing communication protocols are desired not to be changed or impossible to change for a variety of other reasons. In these situations the simple seamless upgrade path described above is not possible, and other methods and processes are needed.

**[0098]** To maintain compatibility with legacy devices in these systems, and to allow usage of communication systems that do not natively include packet or symbol or data sequencing or identification information, the methods introduced herein disclose methods to send the source packet sequence or identifier information in a logically separate stream, so that a

capable receiver can gather this information together with FEC repair data and recover erased or lost or missing or corrupted source packets or symbols. The Source Identification data, as outlined in one embodiment, can capture and represent this information. In some embodiments, this identifier or sequence information is sent in one of the channels that FEC repair data are sent, although this is not absolutely required.

**[0099]** Fig. 2 shows an example of a seamless upgrade path for a communication system 200 that is using a mixture of devices that do and that do not support FEC methods. A Headend device 210 generates and sends to the core network 220 both an original program that comprises original source data, and repair for program that comprises FEC repair data and Source Identification data. The core network 220 sends the original program to Access Networks 230(1), 230(2), does not send the repair program to Access Network 230(1) in which there are legacy set-top boxes (STBs) 240(1), 240(2) that do not support FEC, and does send the repair program to Access Network 230(2) in which there are STBs 250(1), 250(2) that do support FEC methods. There are many variations of the communication system 200, including variations where a legacy STB 240(1) does receive the repair program but only uses the original program, including variations where there is no distinction between a core network and an access network, including variations where the original program and repair program are sent as one logical stream, including variations where retransmission methods are supported instead of or in combination with FEC methods, including variations where set-top boxes are other types of receiving devices, including variations where some receiving devices do not support the Source Identification data methods and some receiving devices support the Source Identification data methods in conjunction with FEC methods and other receiving devices support the Source Identification methods in conjunction with retransmission methods and other receiving devices support the Source Identification methods in conjunction with both retransmission methods and FEC methods and other receiving devices support the Source Identification methods in conjunction with data re-ordering methods, including variations where there are multiple sending devices, including variations where the sending and/or generating device for a repair for program is either logically or physically different from the sending and/or generating device for an original program, including variations where sending devices are at least sometimes receiving devices or vice versa, including variations where the Source Identification data methods are supported for reasons other than FEC methods or retransmission methods, for example to support data re-ordering methods.

**[0100]** It is desirable that a communication systems that supports Source Identification methods and associated recovery methods (such as FEC methods and retransmission methods) has as few differences as possible in terms of signaling and data delivery compared to a communication system that does not support such methods. For example, it is desirable if no special pre-signaling is required to indicate whether a particular stream supports Source Identification methods and associated methods, where pre-signaling might include for example an electronic program guide or its equivalent. For example, for a variety of reasons, some streams of data may support Source Identification methods and associated recovery methods and other streams may not, either at the same time or at different points in time. For example, it might be the case that initially no streams support Source Identification methods, and then over time more and more streams support Source Identification methods and eventually all streams might support Source Identification methods. It can be highly desirable to not have to signal whether streams do or do not support Source Identification methods, for example not having to include in an electronic programming guide whether or not each stream listed in the guide supports Source Identification methods.

**[0101]** Fig. 3 shows an example of possible state diagram 300 that might be used as logic in a receiving device that supports Source Identification methods and FEC methods wherein no pre-signaling is needed to indicate whether a particular stream uses Source Identification methods and FEC methods. In state diagram 300, a receiver starts in the legacy state 310 when first joining a stream, i.e., the receiver initially assumes that the stream is being sent does not use Source Identification methods and FEC methods. Nevertheless, in legacy state 310 the receiver continually monitors reception of packets for the current stream, and if Source Identification data (or FEC repair data) is received then the receiver moves to the SI Un-Sync state 320 in the state diagram. It may be some time, e.g., 100 ms, before the first Source Identification data (or FEC repair data) arrives at the receiver for a stream that uses Source Identification methods and FEC methods, which introduces some latency into moving into the SI Un-Sync state 320. Once the receiver is in the SI Un-Sync state 320, the receiver continually receives both FEC repair data if it is available, as well as Source Identification data that the receiver attempts to use to interpret the identify and/or determine the relative ordering of received and missing source packets or symbols. If the receiver can interpret the current Source Identification data then the receiver moves to the SI Sync state 330. While in the SI Sync state 330 the receiver may use the interpretation of the Source Identification data to identify and/or determine the relative ordering of received and missing source packets or

symbols, and in turn use this to use received FEC repair data to recover parts of the original source stream of data. If the receiver is in the SI Sync state 310 and can no longer interpret the current Source Identification data then the receiver moves to the SI Un-Sync state 320 until the current Source Identification data can be interpreted again. The amount of latency in moving between the SI Sync state 310 and the SI Un-Sync state 320 may be on the order of a protection period, i.e., the period of time it takes to receive on source block of data from the stream, if the source data is organized into source block and is the Source Identification data is organized source block by source block.

**[0102]** There are many variants of state diagram 300. For example, retransmission methods may be used in place of FEC methods or in conjunction with FEC methods. As another example, re-ordering methods may be used in place of FEC methods. As another example, other data may be carried in addition to Source Identification data, for example to indicate whether FEC repair or retransmission methods are in use, or what type of FEC code is being used, or what type of retransmission methods are in use for the stream.

**[0103]** Fig. 4 provides an example of a process flow chart that might be used within a receiving device that supports Source Identification methods and FEC methods. In step 420, the process monitors reception of packets, and when a packet is received the process moves to step 430. In step 430, the process checks to see if SI Sync state is the current state (corresponding to being at SI Sync 330 in Fig. 3). If SI Sync is the current state in step 430 then the process move to step 440. In step 440, the process associates the received packet with the appropriate source block based on Source Identification data, and the process proceeds to step 450. In step 450 the process checks to see if it is time to FEC decode the current source block, and if it is not then the process moves back to step 420 to receive additional packets, but if it is then the process move to step 460 and uses received source and repair packets, and the associated Source Identification data that relates repair packets to source packets, to FEC decode the current source block, and then in step 470 the current source block is output (to for example a media player on the receiving device), the next source block becomes the current source block, and the processing continues on to step 420. If SI Sync is not the current state in step 430 then the process move to step 480. In step 480, the received packet is output (to for example a media player device) if it is a source packet, otherwise it is discarded, and processing returns to step 420.

**[0104]** Many of the details are not shown in Fig. 4. For example, Fig. 4 does not show the reception of Source Identification data that is used to determine the association between source and repair packets and the order of packets, and that is used to determine the status of SI Sync state in step 430. There are many variants of the process shown in Fig. 4. For example, in step 420 multiple packets may be received before proceeding to step 430.

Relationship between packets and symbols

**[0105]** Forward erasure correction codes often operate on symbols chosen from a fixed alphabet. For example, the symbols may be chosen from an alphabet of size $2^M$ in which case there is an obvious mapping between symbols and strings of M binary digits. Such symbols are hereinafter referred to as having a size of M bits. Data is often sent over a communications channel in the form of packets. In some cases, there is a one-to-one mapping between symbols and packets. In other cases, including for example those described in Luby II, packets may be larger than symbols, with a packet comprising several symbols. Additionally, symbols and packets may comprise the same data but the bit order and placement may differ, or additional information may be placed into the FEC symbols which is not contained explicitly within the packet, for example a binary representation of the packet length. Equally, data may be present in packets that are not present in the FEC symbols, for example data that does not need to be protected by the FEC code because it comprises fields that are not used or have the same value in every packet or because it could be recovered in other ways.

**[0106]** Retransmission methods may also be based on symbols, wherein a symbol may be a byte or some large unit of data. For example, a receiver may request a certain byte range from a file or from a stream of data to be resent.

Relationship between source blocks and symbols

**[0107]** Many forward error correction codes operate on discrete collections of source symbols, known as source blocks. For example, a source block may comprise all the source symbols of a file, or all the source symbols of a particular fragment in time of a streaming service.

**[0108]** Each source symbol is given a unique identification within the source block where it belongs. This identification is often not part of the source packet data, but it is often a piece of information used for the FEC decode process.

**[0109]** Retransmission methods may also be based on source blocks, and re-ordering methods may also be based on source blocks. For example, a source block may be of a size that conveniently fits into a receiver memory, and for example it may be convenient to identify a block of data that is brought into memory or written from memory to disk. As another example, a source block may be of a size that corresponds to an amount of data sent over a particular period of time, and for example it may be convenient to distinguish data that is sent during one period of time versus another period of time based on to which source block the data belongs.

**[0110]** In many of the embodiments of methods described herein source blocks are used. For one skilled in the art it should be recognized that source blocks are not an essential ingredient of the methods, and that there are variants of the methods that do not use source blocks.

Receivers

**[0111]** In order for a receiver to utilize the FEC repair data, the receiver often needs to be able to obtain Source Identification data, which may or may not be delivered to the same channel as the FEC repair data. It is usually necessary for the senders and receivers to use a same protocol in representing Source Identification data. An example of such a protocol is given in specific embodiments. The receiver uses Source Identification data to identify positions of received source packets within their source blocks, as well as positions of missing packets. The FEC repair data are then used to recover these missing packets if possible.

**[0112]** During the upgrade process, a legacy receiver can ignore both Source Identification information and FEC repair data, if received. It receives source packets as before the upgrade, and it will continue to perform at pre-upgrade level until being upgraded.

Source Identification Methods

**[0113]** There are many ways to represent and convey Source Identification data. In general there is a trade-off among bandwidth overhead, CPU load, and Quality of Service (QoS).

**[0114]** Uses of the Source Identification method include the ability to distinguish, identify or order among all source symbols or source packets or other source data units to be sent, for example among all source data that is to be sent for a file or stream. There are other cases where the Source Identification methods can be applied so that source data units that are in close proximity to other source data units (for example in the sending order, or sent within a

small window of time) are to be distinguished, identified, or ordered, for example for a stream of data where the amount of reordering, distinguishing, or identifying needs to be only over the maximum amount of source data that is in flight between a sender and a receiver. In these cases, Source Identification methods that partition source data units into multiple source blocks or there equivalent might be appropriate, as methods for doing this are described later within this disclosure. The source block structure used by the Source Identification methods may or may not coincide with the source block structure used by FEC or retransmission methods, if used.

**[0115]** Many of the Source Identification methods described below use hash functions in their descriptions. One of the functions of hash functions is to map long strings to much shorter strings in such a way that distinct long strings are mapped to distinct shorter strings. Equivalently, the hash functions may map values to values. There are many possible hash functions that could be used for the purposes of the methods described herein, for example the hash functions described in the publication entitled "Pairwise Independence and Derandomization", Mike Luby and Avi Wigderson, Foundations and Trends in Theoretical Computer Science, Volume 1, Issue 4, 2005, Print ISSN 1551-305X, Online ISSN 1551-3068 (hereinafter "Hash Functions") could be used. There are many ways to represent a list of hash functions, some of which are described in "Hash Functions", for example by representing the bit sequences that determine how the hash functions are computed on a particular input, or by selecting a small list of hash functions from a family of hash functions and representing the hash function by an index into this list. One example of a possible hash function family can be described by the positive integers (a,b), where a and b range in value between 0 and a positive integer value P, where the hash function determined by the pair (a,b) maps an input X to (a*X + b) modulo P. In this example, a pair of values (a,b) can be used to indicate the selection of the hash function determined by (a,b) if P is fixed or determined by (a,b), or alternatively by the triple (a,b,P) if P is not determined by (a,b). The number of values in the range of a hash function is not restricted to be a prime number, there are well-known methods for hashing similar to those described here that allow the number of values to be any positive number. Another way to indicate a selected hash function from this same family of hash functions (or any family of hash functions) is with respect to a pre-determined list of triples of values $(a_1,b_1,P_1)$, $(a_2,b_2,P_2)$,..., $(a_i,b_i,P_i)$, where an index j can be used to indicate the selection of the hash function determined by $(a_j,b_j,P_j)$.

**[0116]** The description of a first Source Identification method follows. Suppose there are K different source data units (for example source packets or symbols) comprising the source data to be sent. Suppose all K source data units have distinct values. Let H be a hash function that maps the K source data units to K L-bit-vectors, v(0), v(1),..., v(k-1). If H is a random hash function then the probability that two of the K L-bit-vectors map to the same value is approximately K^2/(2^(L+1)), where "^" is the exponentiation function. Thus, to ensure that the K L-bit-vectors are distinct with probability at least 1-eps, the value of L should be approximately 2*log(K)+log(1/eps), where log is the logarithm base 2 operator. As a first example, if K = 1000 = 1e3 and eps = 1e-6, then L should be approximately 40 bits or 5 bytes, where "1e3" is the scientific exponential representation of 10^3 = 1000. As a second example, if K = 1e6 and eps = 1e-12 then L should be approximately 80 bits or 10 bytes. The Source Identification data can be simply the concatenation of the K L-bit vectors, concatenated in the order of the source data units, assuming that a sender and receiver can compute the same hash function H. A sender can compute the Source Identification data from the K source data units by applying the hash function H to each of the source data units to generate the K L-bit vectors. Upon receiving a source data unit, a receiver can apply H to the source data unit to compute the corresponding L-bit vector value, and then from the Source Identification data the receiver can distinguish, identity, and order the received source data unit among the K source data units, i.e., this is an example of a **determining sequencing method**. In the first example above the Source Identification data is 5,000 bytes in size, and in the second example above the Source Identification data is 10,000,000 bytes in size. Both of these might be prohibitive in size for certain applications. For some uses of Source Identification data, for example some retransmission methods, a much shorter Source Identification data may be sufficient at the receiver to identify and request a source data unit. For example, in a NACK based retransmission method, the Source Identification data communicated to a receiver may be the description of the hash function H, and a receiver may compute the hash function H on each received source data unit and send to a sender the L-bit vectors to which the received source data units mapped. In this case, the sender can determine which source data units have not been received by the receiver based on which L-bit vectors are received, i.e., this is an example of a **determining identification labels for known source data units method**. Thus, the amount of Source Identification data communicated to the receiver explicitly may even be zero, assuming that the sender and receiver have pre-established which hash function H to be used. However, in this case a

substantial amount of data in the form of source data unit labels may be communicated from a receiver to a sender in order to request retransmission of missing source data units.

**[0117]** A variant of the above Source Identification method is for the sender to repeatedly choose a new hash function H from a family of hash functions and apply H to the K source data units until hash function H maps the K source data units to distinct vectors. Note that L = 2*log(K) is sufficient for H to map the K source data units to distinct vectors with probability at least ½. Thus, from a list of log(1/eps) random hash functions, there is at least one hash function that will map the K source data units to distinct vectors with probability at least 1-eps. In this case, the Source Identification data can be the concatenation of the index of the hash function H selected by the sender, the value of K and the K L-bit vectors. A receiver can use the Source Identification data to determine the index of the hash function H to use and the value of K, and then upon receiving a source data unit can apply H to that source data unit to determine the L-bit vector that distinguishes, identifies or orders the source data unit among the K source data units, i.e., this is an example of a **determining sequencing method**. If K = 1,000 then the Source Identification data is 20 bits to identify the hash function, the value of K, and 1,000 20-bit vectors, or approximately 2500 bytes, and if K = 1,000,000 then the Source Identification data is 40 bits to identify the hash function, the value of K, and 1,000,000 40-bit vectors, or approximately 5,000,000 bytes.

**[0118]** A third variant of a Source Identification data method follows. A sender can compute the Source Identification data as follows. A first hash function H is chosen to map the K source data units into K distinct L-bit vectors u(1), u(2),...,u(K), for example using the first or second Source Identification data method described above. Let set $S_1$ = {0, 1,...,K-1}. Note that in different variants, the size of $S_1$ may be larger or smaller than K, but in certain embodiments K can be a good choice for the size of $S_1$. Choose a hash function $H_1$ that maps L-bit vectors to $S_1$. Use $H_1$ to map u(1), u(2),...,u(K) to $S_1$. A vector u(i) is said to map uniquely with respect to $H_1$ if there is no j such that u(j) maps to the same value as u(i) with respect to $H_1$, and otherwise u(i) is said to experience a collision with respect to $H_1$. If less than a threshold number of u(1), u(2),...,u(K) map uniquely with respect to $H_1$ then choose a new hash function $H_1$ and repeat the above process until the number of u(1), u(2),...,u(K) that maps uniquely with respect to $H_1$ is at least the threshold number. Note that for any i the probability that u(i) maps uniquely with respect to a hash function $H_1$ that is chosen uniformly at random is approximately 1/e, where e = 2.718281828459045, and thus K/e is a reasonable choice of a threshold number, although other threshold numbers can be

appropriate depending on the value of K, e.g., as K approaches zero the threshold number might approach or reach K. Once an appropriate $H_1$ is determined, initialize a K-bit vector $V_1$ to K zero bits. For every position j where there is an i such that u(i) maps uniquely to position j with respect to $H_1$, set the jth bit of $V_1$ to one.

**[0119]** Let $u_1(1),\ldots,u_1(K_1)$ be the $K_1$ vectors out of the K vectors that experience collisions with respect to $H_1$. Let set $S_2 = \{0,\ldots,K_1\text{-}1\}$and choose a hash function $H_2$ that maps L-bit vectors to $S_2$. Use $H_2$ to map $u_1(1),\ldots,u_1(K_1)$ to $S_2$. If less than a threshold number of $u_1(1),\ldots,u_1(K_1)$ map uniquely with respect to $H_2$ then choose a new hash function $H_2$ and repeat the above process until the number of $u_1(1),\ldots,u_1(K_1)$ that map uniquely with respect to $H_2$ is at least the threshold number, where the threshold value can depend on $K_1$. Initialize a K-bit vector $V_2$ to K zero bits. For every position j where there is an i such that u(i) maps uniquely to position j with respect to $H_2$, set the jth bit of $V_2$ to one.

**[0120]** The above process can be repeated with additional hash functions $H_3,\ldots,H_h$ until all of u(1), u(2),...,u(K) map uniquely with respect to a hash function. In particular, $H_h$ uniquely maps $u_{h-1}(1)$, $u_{h-1}(2),\ldots,u_{h-1}(K_h)$, where $u_{h-1}(1)$, $u_{h-1}(2),\ldots,u_{h-1}(K_h)$ are the $K_h$ vectors out of the K vectors that experience collisions with respect to $H_1,\ldots,H_{h-1}$.

**[0121]** The steps of the third variant of a Source Identification data method are outlined in Fig. 5A. A hash function H is chosen (502) that maps the K source packets (501) to distinct vectors u(1), u(2),...,u(K). Next, a hash function is selected (503) to operate on the subset of the K vectors not yet mapped uniquely by a previous hash function (for the first hash function, the subset is the full set of K vectors). The selected hash function is then used to hash the subset of K vectors (504). During this hashing, information is recorded noting the indices of packets operated, hash functions and resulting values, and potential collisions (506). In one embodiment, this data is recorded in a bit vector. Next, the output of the hash function can be analyzed to determine whether any collisions have occurred (505). If there are collisions, the process is repeated with a different hash function. If the no collisions exist, then the recorded information (506) can be used to form a Source Symbol Map (507).

**[0122]** The Source Identification data or equivalently the Source Symbol Map can be the concatenation of the index of the hash function H, the value of K, the number h of additional hash functions, the indices of the hash functions $H_1,\ldots,H_h$, the values of $K_1,\ldots,K_{h-1}$, and the bit-vectors $V_1,\ldots,V_h$. The position of a source data unit can be determined as follows. Apply hash function H to the source data unit to generate an L-bit label u. Apply $H_1,\ldots,H_h$

consecutively to u until there is a first i such that u is uniquely mapped with respect to $H_i$, and let j be the position to which $H_i$ maps u. The position of the source data unit is the total number of bits in $V_1,\ldots,V_{i-1}$ that are set to 1, plus the number of bits in $V_i$ that are set to 1 within the first j bits. This method for determining the ordering of source data units and their positions is an example of a Source Identification method that enables a determining positioning method, i.e., the positions of the source data units are determined based on the Source Identification data, but the positions are not necessarily the sending positions of the source data units. If K = 1000 and the threshold number K/e is used then the Source Identification data is the bits needed to identify the hash functions and the values of K, $K_1$, $K_2,\ldots,K_{h-1}$, followed by 2700 bits, or approximately 350 bytes. If K = 1,000,000 then the Source Identification data is approximately 340,000 bytes. There are many alternatives and optimizations of this variant possible, as one skilled in the art will recognize, for example the sequence of values $K_1,\ldots,K_{h-1}$ could be chosen differently, the threshold numbers can be chosen differently, the bit-vectors $V_1,\ldots,V_h$ could be multiple bits per entry or a variable number of bits per entry, the last vector $V_h$ can be omitted if it is always the case that all of its bits are set to 1, etc.

**[0123]** Fig. 5B is an illustration of the third variant of the Source Identification data method just described. In the example of Fig. 5B, there are eight data units (510(1),...,510(8)) for which Source Identification data is to be generated. A first hash function H is chosen that hashes the contents of the eight data units (510) to a set of eight L-bit hash values (520), where H is chosen so that if the eight data unit contents (510) are distinct then so are the eight L-bit hash values (520), where in this example the eight L-bit hash values (520) are denoted A, B, C, D, E, F, G and H, respectively. Note that more than one candidate hash function might be chosen before a suitable hash function H is chosen that maps all source data units to unique hash values. Also note that in some embodiment, applying this initial hash function may be skipped, and the process outlines below may process by operating directly on the source data units (510) rather than on the hash values (520).

**[0124]** Referring to Fig. 5B, a second hash function $H_1$ is chosen that maps each of the L-bit hash values (520) to a hash value that ranges between 0 and 7, where the map defined by $H_1$ is indicated by the arrows from the hash values (520) to the hash values (530). Note that more than one candidate hash function might be chosen before a suitable hash function $H_1$ is chosen so that enough of the hash values (520) map to unique hash values (530), where in this example the threshold number might be that at least four of the hash values (520) map to

unique hash values (530). In the bit-vector $V_1$ (540), a value of 1 for bit j indicates that there is a unique hash value (520) that maps to hash value j (530), and a value of 0 for bit j indicates that either zero or two or more hash values (520) map to hash value j (530). In this example, the first bit of $V_1$ is set to 0 because no hash value (520) maps to the value 0 (530) under $H_1$, the second bit of $V_1$ is set to 0 because the hash values B and C (520) both map to the value 1 (530) under $H_1$, the third bit of $V_1$ is set to 0 because no hash value (520) maps to the value 2 (530) under $H_1$, the fourth bit of $V_1$ is set to 1 because the hash value A (520) uniquely maps to the value 3 (530) under $H_1$, the fifth bit of $V_1$ is set to 1 because the hash value G (520) uniquely maps to the value 4 (530) under $H_1$, the sixth bit of $V_1$ is set to 1 because the hash value D (520) uniquely maps to the value 5 (530) under $H_1$, the seventh bit of $V_1$ is set to 0 because the hash values E and H both (520) map to the value 6 (530) under $H_1$, and the eighth bit of $V_1$ is set to 1 because the hash value F (520) uniquely maps to the value 7 (530) under $H_1$. In this example, hash values B, C, E and H (520) do not map to unique values, and thus these four hash values (520) are to be mapped with another hash function $H_2$ to four values 0 through 3 (550), resulting in a collision bit-vector $V_2$ (560). In this example, the hash values E and H (520) are mapped uniquely by $H_2$, whereas the hash values B and C (520) collide under the $H_2$ mapping, and thus the two hash values B and C (520) are to be mapped by another hash function $H_3$ to two values 0 through 1 (570), resulting in a collision bit-vector $V_3$ (580), and in this example all remaining values are mapped uniquely by $H_3$ and thus $H_3$ is the last hash function needed to complete the Source Identification data. The Source Identification data in this example comprises the value h=3, the values K=8, $K_1$=4, $K_2$=2, the representation of the hash functions H, $H_1$, $H_2$, $H_3$, and the bit vectors $V_1$, $V_2$, $V_3$.

**[0125]** In the example shown in Fig. 5B, the indices just to the left of the collision bit-vectors $V_1$,$V_2$, and $V_3$ indicate the positions of the source data units so determined by this Source Identification data, i.e., position 0 corresponds to the packet with sequence number 0, position 1 corresponds to the packet with sequence number 6, position 2 corresponds to the packet with sequence number 3, position 3 corresponds to the packet with sequence number 5, position 4 corresponds to the packet with sequence number 7, position 5 corresponds to the packet with sequence number 4, position 6 corresponds to the packet with sequence number 2, and position 7 corresponds to the packet with sequence number 1. From this Source Identification data, a client can easily compute the positions of any received or missing of the eight source packets. Thus, for example, the positions could be used to order the source

packets within a source block for FEC encoding and decoding purposes, and thus the Source Identification data is sufficient for FEC encoding and decoding of source data units. As another example, a receiver may compute the positions of all received source data units, and from this determine the positions of missing source data units. A receiver can send requests for missing source data units, identifying which source data units are being requested via their positions, and a sender can determine which source data units to send in response based on the requested positions. This third variant is an example of a Source Identification method that enables a **determining positioning method**.

**[0126]** As a fourth variant, the Source Identification data of the third variant can be augmented to include mappings between the sequencing of the K source data units (where for example their sequencing corresponds to the original source data order or as another example to the source data sending order) and their corresponding positions described above for the third variant. For example, the additional sequencing data can comprise an ordered list SEQ = $<seq_0, seq_1,\ldots, seq_{K-1}>$ of K sequence numbers, where $seq_j$ is the sequence number of the data unit that has position j. This additional sequencing data can thus comprise a position to sequence mapping.

**[0127]** An example of the Source Identification data for this fourth variant is shown in Fig. 6A, which is an augmentation of the example shown in Fig. 5B. For the example of Fig. 5B, K=8 and h=3. As shown in Fig. 6A, the h-value is 3 (610), the K-values are K=8, $K_1$=4, $K_2$=2 (620), the four hash functions are H, $H_1$, $H_2$, $H_3$ (630), the bit vectors $V_1$, $V_2$, $V_3$ of respective sizes 8, 4 and 2 are shown concatenated in (640), and the additional sequencing data comprising K=8 sequence numbers is shown in (650). In this case, the source data unit that is mapped to position 0 according to the hash functions is the source data unit with sequence number 0, the source data unit that is mapped to position 1 has sequence number 6, the source data unit that is mapped to position 2 has sequence number 3, the source data unit that is mapped to position 3 has sequence number 5, the source data unit that is mapped to position 4 has sequence number 7, the source data unit that is mapped to position 5 has sequence number 4, the source data unit that is mapped to position 6 has sequence number 2, and the source data unit that is mapped to position 7 has sequence number 1.

**[0128]** For this fourth variant, Source Identification data is sufficient to distinguish, identify and determine the order of the K source data units, i.e., this is an example of a Source Identification method that enables a **determining sequencing method**. The additional size

of the Source Identification data for this fourth variant is K*log(K) beyond the size of the Source Identification data in the third variant. For example, if K = 1,000 then the additional Source Identification data size beyond that of the third variant is 1250 bytes, and if K = 1,000,000 then the additional size is 2,500,000 bytes. In some implementations of this fourth variant, the portion of the Source Identification data that corresponds to the third variant can be sent to a receiver that enables determining positions of source data units, this can be used to FEC decode the source block (where the ordering of the source data units within the source block are according to their positions), and the additional portion of the Source Identification data that enables determining sequencing of source data units is included within the source block that can be recovered using FEC decoding methods. For example, referring to Fig. 6A and Fig. 6B, the h-value (610), the K-values (620), the hash function descriptions (630) and the bit-vectors (640) can comprise the Source Identification data that is communicated to a receiver in order to determine the positions of the source data units in a source block. This is Source Identification data can be in the form of a Source Symbol Map (601). The information contained in the Source Symbol Map can be parsed (602) to extract the data outlined in Fig. 6A except not including the position-to-sequence mapping (650). (This is also illustrated at 603.) The Source Symbol Map can then be used in a determining positioning method, involving a number of hash calculations and collision resolutions (605), on the received source data units. In one embodiment, the received source data units can be in the form of source packets (604). The received symbols can be validated and associated with positioning information (606) before being fed to an FEC decoder (607). The position-to-sequence mapping data (650) can comprise the Source Identification data that is included in a source block that is available after FEC decoding and used to determine the sequencing of the recovered source data units (608). Fig. 7 shows an example of position-to-sequence mapping data (650) included in a source block comprising a number of Source Data Units (510(1-8)). In Fig. 7, the position-to-sequence mapping data (650) is appended to the end of the Source Data Units (510(1-8)) included in the source block where the order of the source data units in the source block is determined by the position information.

**[0129]** As another variant, the Source Identification data can be further augmented to include the size of each source data unit. This variant is generally not used if all source data unit are the same or approximately the same size. This variant enables the positions of variable size source data units to be more precisely determined.

**[0130]** Fig. 8A is a example of another embodiment of Source Identification data similar to the example in Fig. 5B. One difference between the example shown in Fig. 5B and the example shown in Fig. 8A is that the Bit-Vectors in Fig. 8A include two bits of information rather than just one. This extra data can be used to communicate more details on any collisions, or lack thereof, for the hashing operation at a given level. For example, in Fig. 8A, "11" is used to indicate that none of the hashed values mapped to a particular location in the Bit-vector; "00" is used to indicate that a particular location in the bit vector maps to a unique source packet, and "10" is used to indicate that more than one source packet maps to a particular location. One skilled in the art will recognize that other similar schemes can be used to communicate information such as this.

**[0131]** Fig. 8B illustrates an example of Source Identification information formatted in a data structure (810). The data structure can then be used in a variety of ways. One skilled in the art will recognize that there are many possible formats for the Source Identification information. Any data structure used to organize the Source Identification information can be inserted into each packet or spread across multiple packets that may be optionally FEC protected that is then transmitted to a receiver.

**[0132]** In the data structure (810) embodiment illustrated in Fig. 8B, the approximate size of each of the illustrated fields is listed below the field in bytes. The size of the fields may vary between difference embodiments of the data structure. in Fig. 8B, G represents the number of symbols per packet. If G equals 0, this indicates that there are variable length packets. H represents the number of modulo hash attempts used to create the Source Identification information. F is the initial hash function index. Check is a checksum of a source block. B is the hash position map bit allocation. L is the hash collision vector lengths for each iteration of the hashing. In Fig. 8B, L is 2 bytes each for first 2 levels, 1 byte each thereafter. V stores the actual hash collision vectors. M stores the hash position map. In Fig 8B, each value takes B bits. B1 is an optional field that is only applicable with G = 0. B1 stores the bit allocation for variable packet symbol number. M1 is another optional field that is only applicable with G = 0. M1 stores the number of symbols in the packets.

**[0133]** Fig. 8B also illustrates an example (820) of this general format using data from Fig. 8A.

How a receiver uses the Source Identification data.

**[0134]** The protocol communicates between senders and receivers the number of hashed packets $\{K, K_1, \ldots, K_n\}$, collision vectors $\{V, V_1, \ldots, V_n\}$, and hash functions $\{H, H_1, \ldots, H_n\}$, where $H_n$ results in no collision, i.e., $V_n$ is all 0's. In addition, we also communicate the mapping between the positions of all K source packets and their corresponding hashed position within vectors $\{V, V_1, \ldots, V_n\}$, as well as the number of symbols in each source packet.

**[0135]** As an example of how receiver can determine the sequence number for a received source data, suppose a receiver receives the fifth source data unit (510(5)) according to the example shown in Fig. 5B. The receiver applies H to this source data unit to obtain the hash value E (520). The receiver applies $H_1$ to E (520) to obtain 6 (530). Since $V_1$ at position 6 is equal to 0 (540), the receiver determines that there is a collision of E (520) with respect to $H_1$ with respect to some other hash value (520), and thus applies $H_2$ to E (520) to obtain 3 (550). Since $V_2$ at position 3 is equal to 1 (560), the receiver determines that E (520) maps uniquely with respect to $H_2$. The receiver can determine the position of this source data unit determined by the hash functions by counting the number of 1's in $V_1$ and adding that to the number of 1's in $V_2$ up to but not including the 1 in position 3 of $V_2$, which is equal to 5 and thus the position of the fifth source data unit is 5. Referring to Fig. 6A, the receiver can determine the sequence number of the fifth source data unit by looking in position 5 of the position-to-sequence mapping (650), which is equal to 4 and thus the sequence number of the fifth data unit is 4. By using similar processes for each received source data unit, the receiver can determine the sequence number of each received source data unit, and since the sequence numbers are consecutive starting at 0 and proceeding up to K-1, this allows the receiver to determine the sequence numbers of all missing source data units.

**[0136]** One potential difficulty is the identification of a particular source block when the Source Identification data is used to identify a portion instead of the entire source data, as the source packets themselves may not contain that information, and there may not be an existing synchronization protocol for determination and communication of source block boundary. One possible solution is to use the above Source Identification communication protocol over a wider range of source packets than the source block of interest, i.e., the generated Source Identification data may overlap with previous and next source block. An example of this solution is illustrated in Fig. 9. The source block boundary may be signaled and

communicated with respect to its relative position within the entire range of source packets involved. The wider the range of overlap, the more robust this protocol becomes, with the cost of extra bandwidth overhead as well as CPU load.

**[0137]** Fig. 10 illustrates an example of one embodiment, similar to the example shown in Fig. 8B, but where the structure of the Source Identification data has been modified to include a new field, "O," that represents the offset from reference. The offset from reference, as illustrated in Fig. 9, can determine the amount of overlap in the generated Source Identification data between source blocks. This offset can be used to determine the overlap of source data units at both the beginning and at the end of the source block with other adjoining source blocks, respectively. Any packets received within this offset range (that below to adjoining source blocks) can be discarded with respect to determining the sequencing and positioning of source data units within the source block covered by this Source Identification data. (Other Source Identification data that covers adjoining source blocks to which these discarded packets belong can handle determining how these packets are sequence or positioned within these other adjoining source blocks.)

**[0138]** Another potential problem is that two or more source packets within a single source block may be identical. These identical packets will always have hash collision regardless of the hash function applied to the packets. In the hash position map, both packet sequences will map to the same hash position. This is illustrated in Fig. 11A.

**[0139]** A method can be devised that indicates which source packets are identical and communicates this piece of information to the receiving device. This method thus reduces the problem for the receiver to just dealing with source packets that are all different from each other, in which case the aforementioned protocol can be applied. For our communications purpose, a collision resulting from identical packets is NOT marked in the collision bit vectors assuming that the two packets are identical and that they don't collide with any other different packets in a hash function. The same output value implies the existence of identical source packets. An example of such a method is the following. Instead of using a position-to-sequence mapping, for example (650) as shown in Fig. 6A, use a sequence-to-position mapping. Then, each identical packet, independent of its sequence number, maps to the same position, and for each identical packet the sequence-to-position map can list at its sequence number the same position to which it maps. For example, suppose there are nine source data units (1110) as shown in Fig. 11B with corresponding hash values from H shown (1120),

which are the same hash values (520) shown in Fig. 5B. Suppose the positioning determined by these hash values (1120) is the same as shown in Fig. 5B. Then, the sequence-to-positioning mapping (1130) shows that the identical source data units with sequence numbers 4 and 8 are both mapped to the source data unit in position 5. When using this method, it might be useful to indicate both the number of source data units in the sequence and the number of unique source data units as part of the Source Identification data, e.g., 9 and 8 in the example of Fig. 11B.

**[0140]** Another potential problem is that there may be rare cases that different packets will still collide after every available hash function has been applied to the packets. This can happen due to limited hash function selection pool, or due to insufficient amount of time allocated to resolve the collision. In these cases, the last collision bit vector will have one or more bits marked for "final collision". The receiver will simply discard the packets whose hash collision cannot be resolved. This potential solution is illustrated in Fig. 12.

**[0141]** The use of Source System Maps to identify source data offers many advantages. For example, Source System Maps are reasonably efficient in size since they only require about an additional 4 to 12 bits of data for each source packet for some of the embodiments described for some values of K ranging up to a couple of hundred. Also, multiple hash function help to reduce CPU load. Low complexity hash functions are operated first; higher complexity hash functions then operate later on many fewer packets. Another advantage is that no additional method is needed to resolve hash collision. Also, Source System Maps are flexible to hash function design or replacement.

Reliable Transmission of Source Identification Data

**[0142]** Depending on network configuration, the Source Identification data may be either combined with FEC repair data in a single channel, or independently transmitted. Since this piece of information can be crucial in packet recovery and/or reordering process, it is preferred to apply forward erasure correction techniques on the Source Identification data. Note that the FEC applied on the Source Identification data does not have to bear any relationship with the FEC code on the original source stream. It may or may not be the same FEC code, it may or may not be systematic. In addition, spreading the transmission of these Source Identification data packets (and their FEC repair part) can be beneficial because the transmission is then more resilient to burst erasure. Another technique is to combine all or parts of the Source Identification data with the FEC repair data generated from the original

source stream, if possible, in order to further reduce bandwidth overhead. In cases where the Source Identification data is not used with FEC repair data of the original data, it can still be preferable to FEC protect the Source Identification data.

**[0143]** In one embodiment, the source symbol map can be transmitted along with FEC data (such as data generated using a chain reaction code). The result is that a receiver can receive two streams - one stream with unmodified source data, and a second stream with FEC data for the original stream (chain reaction repair data) and Source Symbol map data. The source symbol map data can in turn also have its own FEC data.

**[0144]** Fig. 13 shows one example of a method that generates FEC repair data from a Source Symbol Map. In Fig. 13, the Source Symbol Map is first broken in a number of fragments (1310) that are then used as the input to an appropriate FEC Protection scheme (1320). The FEC Protection scheme generates a set of repair data (1330) that can then be transmitted along with the original Source Symbol Map. In one embodiment, sequence numbers are applied to the fragments of the Source Symbol Map (1310) and the repair data (1330) so that if there are J Source Symbol Map fragments, then the Source Symbol Map fragments are assigned sequence numbers ranging from 0 to J-1, and the repair data is assigned to sequence numbers beginning with J and progressing upwards from J.

**[0145]** Fig. 14 is an illustration of repair symbols being generated from source packets. In one embodiment, the repair symbols are generated using a chain reaction code. As referenced in Fig. 14, source packets may contain multiple source symbols and source packet may each contain a different number of symbols.

**[0146]** Fig. 15 illustrates the structure of a repair packet information in one embodiment wherein the repair packet contains both FEC data and Source Identification data. In the embodiment shown in Fig. 15, the Source Identification data is in the form of a source symbol map. In Fig. 15, R represents the number of FEC repair symbols in the packet. In a rare case R may be 0 if there are no repair symbols in the packet. SBN represents the Source Block Number that is protected by this packet. In the embodiment shown in Fig. 15, SBN can range from .0 – 4095. PP is the protection period provided by the FEC scheme. In the embodiment illustrated in Fig. 15, PP is in units of 10ms. SBN and PP can both be used to enhance the synchronization of the stream. ESI is the Encoding Symbol ID of first FEC repair symbol in the packet. K is the number of source symbols in the source block. J is the SSM fragment sequence number. X is the number of original SSM fragments. Consecutive

FEC repair symbols within the same packet have consecutive encoding symbol IDs. The SSM fragment can comprise either original Source Identification data or FEC repair data for an original Source Identification data fragment.

**[0147]** Fig. 16 illustrates one example embodiment showing how a Source System Map can be reconstructed using the fields from the data structure illustrated in Fig. 15. In Fig. 16, fields J, X, and the SSM fragment from the repair packet (1610) are expanded to show what SSM data may be carried in FEC repair packets protecting the original source data in one embodiment.

**[0148]** Fig. 17 is a flow chart illustrating how a receiver can use a Source System Map to apply to Source Symbols received by the receiver. The SSM information is applied (1720) to received source packets (1710). If the receiver is unable to identify a unique position for the source packet or if the process outputs an invalid result for any reason (1730), for example the source packet is not within the valid range of positions or sequence numbers for this source block, the source packet can be discarded (1740) from this source block. If the mapping is successful, Source Symbol positions and/or sequencing can be extracted from the Source Symbol Map (1750) and associated with the source symbols. The source symbols can then be passed onto a FEC decoder, such as a chain reaction decoder (1760), for further processing, such as recovery of the source block.

Transmission Error Handling

**[0149]** In many embodiments, the data streams transmitted from an encoder to a decoder are transmitted over a network in a series of packets. In many such networks, there are many potential errors that can occur in the transmission of the packets. Packets can be lost, packets delivery can be delays, and packets may be duplicated during transmission and the decoder may receive identical packets at different times. In some embodiments, a decoder is subject to time-constraints and may time-out the decoding of a given source block after a period of time. However, the nature of the network may result in packets being delivered to a receiver after the relevant source block has already timed-out. Below are some examples of how a receiver can treat various packet errors. None of the examples below are meant to give an exhaustive list of how various errors can be handled by a receiver.

**[0150]** Fig. 18 illustrates a packet reception time-out wherein the lost packets span two blocks. If it is possible that source packets can be associated with the wrong block, both blocks can be flushed to the application. As used herein, flushing can mean that no attempt is

made to decode those blocks; the source packets are returned as is. Alternatively, flushing can mean that the source packets are simply discarded. Operation for block N+2 will return to normal receiving and decoding. Occasionally the time-out may be associated with lost packets spanning more than two blocks. In such cases, the source packets in between can be stored in a temporary block and then ultimately can be flushed to the application.

**[0151]** Fig. 19 illustrates early burst loss early in a source block. The packets that belong to block N are flushed to the application. The operation on block N+1 can proceed with normal receiving and decoding.

**[0152]** Figs. 20 and 21 illustrate delayed packets that are received either within the same block or within an extended block. In both situations, the delayed packets can be correctly identified within block N with no special handling. Receiving and decoding operations proceed as normal for source block N and for source blocks after N.

**[0153]** Fig. 22 illustrates a delayed packet that crosses beyond an extended block. Block N will be decoded as usual. For block N+1, there are three possible scenarios for what can happen to the delayed packet. The first possibility is that the delayed packet is deemed invalid. For example, the packet may not correctly hash to any packet ID contained in the SSM. In this case, the packet is discarded and block N+1 will be decoded as normal. The second possibility is that the delayed packet is assigned a packet ID according to the SSM on block N+1, but this assigned packet ID coincides with the ID of another packet received for block N+1. In this scenario, no attempt is made to decode block N+1 and block N+1 is flushed to the application. The third possibility is that the delayed packet is assigned a packet ID according to the SSM on block N+1 and no other packet in block N+1 assumes this packet ID. In this scenario, decoding is attempted on block N+1 and the checksum of recovered block is relied upon to invalidate this decoding attempt. The invalidation results in returning to app only received source packets for block N+1.

**[0154]** Fig 23A/23B illustrate a duplicated packet received within extended block. In both situations, the duplicated packets can be correctly identified within block N. Receiving and decoding operations can easily proceed as normal.

**[0155]** Fig. 24 illustrates a duplicated Packet received crossing beyond an extended block. Block N will be decoded as usual. For block N+1, there are three possible scenarios for what can happen to the duplicated packet. The first possibility is that the duplicated packet is deemed invalid.. For example, the packet may not correctly hash to any packet ID contained

in the SSM. In this case, the packet is discarded and block N+1 will be decoded as normal. The second possibility is that the duplicated packet is assigned a packet ID according to the SSM on block N+1, but this assigned packet ID coincides with the ID of another packet received for block N+1. In this scenario, no attempt is made to decode block N+1 and block N+1 is flushed to the application. The third possibility is that the duplicated packet is assigned a packet ID according to the SSM on block N+1 and no other packet in block N+1 assumes this packet ID. In this scenario, decoding is attempted on block N+1 and the checksum of recovered block is relied upon to invalidate this decoding attempt. The invalidation results in returning to the application only received source packets for block N+1.

Case Studies

**[0156]** Below are a number of case studies showing characteristics of the described methods on a variety of different systems. These are case studies of streaming applications where an FEC code is used to add FEC repair packets to protect the original stream, and where each SSM covers a small portion of the entire data stream. In all of these case studies,

- Each source data packet carries 7 MPEG-2 TS units, where each MPEG-2 TS unit is 188 bytes.
- The symbol size is chosen to be 1320 bytes in order that each source packet is one symbol. Each repair packet carries one FEC repair symbol and other data such as all or parts of an SSM.
- The SSM is either carried in whole in each FEC repair packet or is carried in parts within FEC repair packets.
- Each SSM covers one source block of data to which FEC protection is to be applied, and each SSM carries some overlapping information about adjoining source blocks in order to filter out packets from those source blocks.
- In each case a maximum packet size of 1500 bytes is desired in order to avoid packet fragmentation when sending packets over a network.

In each case, two values are derived: $P_{SSM}$ and $P_{FEC}$. $P_{SSM}$ is the probability that not enough packets arrive for the source block to recover the SSM data, whereas $P_{FEC}$ is the probability that not enough packets arrive for the source block to recover the source block assuming an

ideal FEC code. These probabilities are calculated assuming random and independent packet loss under the loss conditions stated for each case.

**[0157]** Case Study 1

- The source data stream rate is 4 Mbps.
- SSM coverage period (= protection period): 125 ms
- Loss conditions: 0.1%
- Derived parameters:
  - Number of packets in a source block = 48
  - Number of repair packets = 3
- SSM considerations:
  - Number of packets used for SSM (5 packet overlap on each end) = 58
  - Number of hash functions = 5
  - Hash collision vector allocation $\approx$ 20 bytes
  - Hash position map allocation $\leq$ 58 bytes
  - The SSM length is less than 100 bytes, and thus the entire SSM can be carried in each FEC repair packet.
- Failure Rates:
  - $P_{SSM} = (0.1\%)^3 = 10^{-9}$
  - $P_{FEC} = 2.4 * 10^{-7}$

**[0158]** Case Study 2

- The source data stream rate is 4 Mbps.
- SSM coverage period (= protection period): 250 ms
- Loss conditions: 0.7%
- Derived parameters:
  - Number of packets in a source block = 95

- Number of repair packets = 9

› SSM considerations:

- Number of packets used for SSM (10 packet overlap on each end) = 115
- Number of hash functions = 5
- Hash collision vector allocation ≈ 40 bytes
- Hash position map allocation ≈ 114 bytes
- Entire SSM length ≈ 168 bytes
- The SSM is split into 2 segments
- FEC protection is used to generate another 9-2=7 repair SSM segments
- Each FEC repair packet carries one SSM segment

› Failure Rates:

- $P_{SSM} = 5.156 * 10^{-17}$
- $P_{FEC} = 4.052 * 10^{-9}$

**[0159]** Case Study 3

› The source data stream rate is 6 Mbps.

› SSM coverage period (= protection period): 500 ms

› Loss conditions: 0.7%

› Derived parameters:

- Number of packets in a source block = 285
- Number of repair packets = 25

› SSM considerations:

- Number of packets used for SSM (30 packet overlap at each end) = 345
- Number of hash functions = 7
- Hash collision vector allocation ≈ 120 bytes

- Hash position map allocation ≈ 400 bytes
- Entire SSM length ≈ 538 bytes
- The SSM is split into 5 segments
- FEC protection is used to generate another 25-5=20 repair SSM segments

Each FEC repair packet carries one SSM segment

› Failure Rates:

- $P_{SSM} = 6.879 * 10^{-42}$
- $P_{FEC} = 1.887 * 10^{-15}$

**[0160]** Fig. 25 is a failure rate chart for a 4Mbps stream for varying protection periods and protection amounts under 0.7% loss conditions. A conclusion that can be drawn from this data is that the probability of decoding failure due to incomplete SSM is far less than that due to insufficient number of FEC received symbols.

Multiple Stream/Retransmission Embodiment

**[0161]** Fig. 26 shows an example of a communication system 2600 that is using a mixture of streams to send data to a set-top box. A Headend device 2610 sends to network 2620 an original program that comprises original source data in a source stream, and two separate repair streams for the program that can comprises FEC repair data and/or Source Identification data. The network 2620 sends the original program to an intermediate server 2630. This intermediate server 2630 sends the source stream and one of the repair streams to a network 2650. Network 2650 may be the same network as network 2620, but it may also be a distinct network. In any event, network 2650 then sends the source stream and the one repair stream to one or more STBs 2640. In the event that the data received by any of the STBs is insufficient to create the source data, the STBs can request retransmission stream. This request is sent to the intermediate servers 2630 that can respond to a request for retransmission by sending a second repair stream that is distinct from the first repair stream. In some embodiments, packets from the second repair stream is cached on the intermediate servers. In some embodiments, the intermediate servers 2630 cached repair data that has not already been sent to a STB. In some embodiments, the intermediate servers 2630 cache repair data based on a protection period of the source stream. In some embodiments, caching of source or repair streams may occur at other locations in the system/network. In some

embodiments, the STBs can request a specific amount of repair data in its retransmission request, such as a specific number of repair packets. In some embodiments, the STBs can request repair data (source data or FEC repair data) based on Source Identification data the STBs have received. In some embodiments, STBs that initially receive source data and/or repair data first receive data that has been cached at the intermediate servers in a unicast stream(s), and then the intermediate server transitions the stream(s) sent to the STBs to a multi-cast stream(s). There are many variations of the communication system 2600, including variations where the source stream and repair streams are sent as one logical stream, including variations where set-top boxes are other types of receiving devices, including variations where some receiving devices do not support the Source Identification data methods and some receiving devices support the Source Identification data methods in conjunction with FEC methods and other receiving devices support the Source Identification methods in conjunction with retransmission methods and other receiving devices support the Source Identification methods in conjunction with both retransmission methods and FEC methods and other receiving devices support the Source Identification methods in conjunction with data re-ordering methods, including variations where there are multiple sending devices, including variations where the sending and/or generating device for a repair for program is either logically or physically different from the sending and/or generating device for an original program, including variations where sending devices are at least sometimes receiving devices or vice versa, including variations where the Source Identification data methods are supported for reasons other than FEC methods or retransmission methods, for example to support data re-ordering methods.

DVB-T/DVB-H Embodiment

**[0162]** Source Identification data transmitted in a data stream separate from a data stream containing the source data can be applied to a variety of systems. For example, Appendix A discloses a system wherein FEC protection and Source Identification data can be transmitted on a first data stream carried on a network such as a DVB-T broadcast network while the original source data can be transmitted on a second data stream that is transmitted on the same network. Appendix A is incorporated into this disclosure for all purposes. In other embodiments, the FEC protection and Source Identification data can be transmitted first data stream carried on a network such as a DVB-H broadcast network while the original source data can be transmitted on a second data stream that is transmitted across another network such as a DVB-T network

**[0163]** Note that DVB-T networks and settings have been designed to transmit to fixed position receivers, whereas there are great advantages to extending the usage of DVB-T networks to other types of receivers, e.g., mobile receivers or fixed position receivers in worse positions. For these types of new receivers, the error-conditions present on existing DVB-T networks are often too challenging to support relevant applications such as high quality streaming video. Using the methods described in this disclosure, the usage of these networks can be extended to these new receivers to support applications that are too challenging to otherwise support. No network upgrade is required for these methods to apply, nor is there a requirement to upgrade existing receivers to support these new receivers. Using the methods described in this disclosure, new receivers will be able to receive the original unmodified data streams, and in addition to receive Source Identification data and FEC repair data associated with these original data streams in order to recover and play-out video streams at the same quality as existing receivers, even though the network conditions are much worse for the new receivers than they are for existing receivers.

**[0164]** As will be clear to those of skill in the art upon review of this disclosure, the methods described herein can be naturally extended to support different FEC codes, maintaining the property that, at the receiver, data from more than one transmitted code can be combined in a way that provides for greater error correction than if the data were processed independently according to the procedures associated with the respective codes.

**[0165]** The foregoing description has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed, and obviously many modifications and variations are possible in light of the above teaching. The described embodiments were chosen in order to best explain the principles of the invention and its practical application to thereby enable others skilled in the art to best utilize the invention in various embodiments and with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the claims appended hereto.

WHAT IS CLAIMED IS

1. A method for generating and transmitting source identification data for source data, wherein the source data has been divided into K source data units, wherein K is greater than one, wherein the source identification data can be used to identify the position of source data units relative to other source data units and to identify lost or corrupted source data units, the method comprising:

generating source identification data from the K source data units;

transmitting a first data stream to a receiver, wherein the first data stream comprises a set of the source data units; and

transmitting a second data stream to the receiver, wherein the second data stream comprises the source identification data.

2. The method of claim 1 wherein the source identification data enables a recipient of the first and the second data streams to determine identification labels for the K source data units used to generate the source identification data.

3. The method of claim 1 wherein the source identification data enables a recipient of the first and the second data streams to determine identification labels for any source data unit.

4. The method of claim 1 wherein the source identification data enables a recipient of the first and the second data streams to determine a position of a given source data unit with respect to an ordering of all the source data units.

5. The method of claim 1 wherein the source identification data enables a recipient of the first and the second data streams to determine a position of a given source data unit with respect to a source-determined order sequence of all of the source data units.

6. The method of claim 1 wherein the second data stream further comprises repair data for the source data units.

7. The method of claim 1 wherein the second data stream further comprises repair data for the source identification data.

8. The method of claim 1 wherein the first data stream is transmitted via DVB-T and wherein the second data stream is transmitted via DVB-H.

9. The method of claim 1 further comprising:

receiving a request from the receiver for additional data; and

transmitting a third data stream to the receiver.

10. The method of claim 9 wherein the third data stream comprises a second set source data units.

11. The method of claim 9 wherein the third data stream comprises repair data for the source data units.

12. The method of claim 1 wherein generating source identification data from the K source data units comprises:

creating an ordered set S from each of the K source data units;

i) applying a hash function, $H_i$, to each member of S, wherein $H_i$ maps the members of S to one of a possible L hash outputs and wherein $H_i$ maps at least X of the members of S to a unique hash output;

ii) creating a vector corresponding to the L hash outputs, wherein the vector indicates which of the L hash outputs experience collisions and which of the L hash outputs uniquely map to a member of S;

iii) creating a set, $S_i$, comprising the elements of S that do not map to a unique hash output;

iv) iterating steps i-iii until either $S_i$ is an empty set or a maximum number of iterations have been executed, wherein for each iteration of steps i-iii $S_i$ is used as S, wherein for each iteration $H_i$ is a hash function that differs from the hash function used during the previous iteration of steps i-iii, wherein for each iteration the values of L and X can change; and

generating source identification data from the vectors created during each iteration of steps i-iii.

13. The method of claim 12 wherein the vector further indicates the location of each uniquely mapped member of S in S, the method further comprising:

generating a position-to-sequence mapping from the location of the uniquely mapped members of S as indicated by the vectors created during each iteration of steps i-iii.

14. A computer readable medium comprising code for performing the method of claim 1.

15. A transmitter comprising a processor and the computer readable medium of claim 14 coupled to the processor.

16. A method for receiving source identification data and associating the source identification data with source data, wherein the source data has been divided into K source data units, wherein K is greater than one, wherein the source identification data can be used to identify the position of source data units relative to other source data units and to identify lost or corrupted source data units, the method comprising:

receiving a plurality of source data units from a first data stream;

receiving source identification data from a second data stream; and

associating the source identification data with the plurality of source data units.

17. The method of claim 16 further comprising:

determining identification labels for the received plurality of source data units.

18. The method of claim 16 further comprising:

determining identification labels for any source data units that are not a member of the received plurality of source data units.

19. The method of claim 16 further comprising:

determining a position for a source data unit in the plurality of source data units with respect to an ordering of all the source data units.

20. The method of claim 16 further comprising:

determining a position for a source data unit in the plurality of source data units with respect to a source-determined order sequence of all of the source data units.

21. The method of claim 16 further comprising:

receiving repair data for the plurality of source data units from the second data stream.

22. The method of claim 16 further comprising:

receiving repair data for the source identification data from the second data stream.

23. The method of claim 16 wherein the first data stream is received via DVB-T and wherein the second data stream is received via DVB-H.

24. The method of claim 16 further comprising:

analyzing the source identification data and the plurality of source data units to determine if any of the K source data units have not been received; and

if any of the K source data units have not been received, requesting additional data.

25. The method of claim 24 further comprising:

receiving the additional data from a third data stream.

26. The method of claim 16 wherein associating the source identification data with the plurality of source data units comprises:

creating a set, S, from each of the plurality of source data units;

i) applying a hash function, $H_i$, to each member of S to obtain a hashed output associated with each of member of S, wherein $H_i$ maps each of the members of S to one of a possible L hash outputs;

ii) determining which of the hashed outputs uniquely identify the source data unit associated with the hashed output;

iii) if it is determined that a hashed output does uniquely identify a source data unit, removing the uniquely identified source data unit from S and associating a proper set of identification data from the source identification data with the uniquely identified source data unit; and

iv) if it is determined that one of the hashed output is not unique, iterating steps i-iii unit S is an empty set or until a maximum number of iterations have been executed, wherein for each iteration $H_i$ is a hash function that differs from the hash function used during the pervious iteration of steps i-iii with a potentially different value of L.

27. The method of claim 26 wherein the $H_i$ applied for each iteration of steps i-iv is determined from the source identification data.

28. The method of claim 26 wherein determining which of the hashed outputs uniquely identify the source data unit associated with the hashed output comprises:

comparing each hashed output to a vector from the source identification data, wherein the vector corresponds to the possible L hash outputs, and wherein the vector indicates which of the L hash outputs experience collisions and which of the L hash outputs uniquely map to a member of S.

29. The method of claim 28 wherein the source identification data includes a position-to-sequence mapping and wherein associating a proper set of identification data from the source identification data with the uniquely identified source data unit comprises associating location information from the position-to-sequence mapping to the uniquely identified source data unit.

30. The method of claim 26 further comprising:

if a source data unit is associated with a proper set of identification data from the source identification data, passing the source data unit to a decoder.

31. The method of claim 26 further comprising:

if a source data unit is not associated with a proper set of identification data from the source identification data, discarding the source data unit.

32. A computer readable medium comprising code for performing the method of claim 16.

33. A receiver comprising a processor and the computer readable medium of claim 32 coupled to the processor.