



DF Raptor Technology

Advanced FEC Technology for Streaming Media and Data Distribution Applications

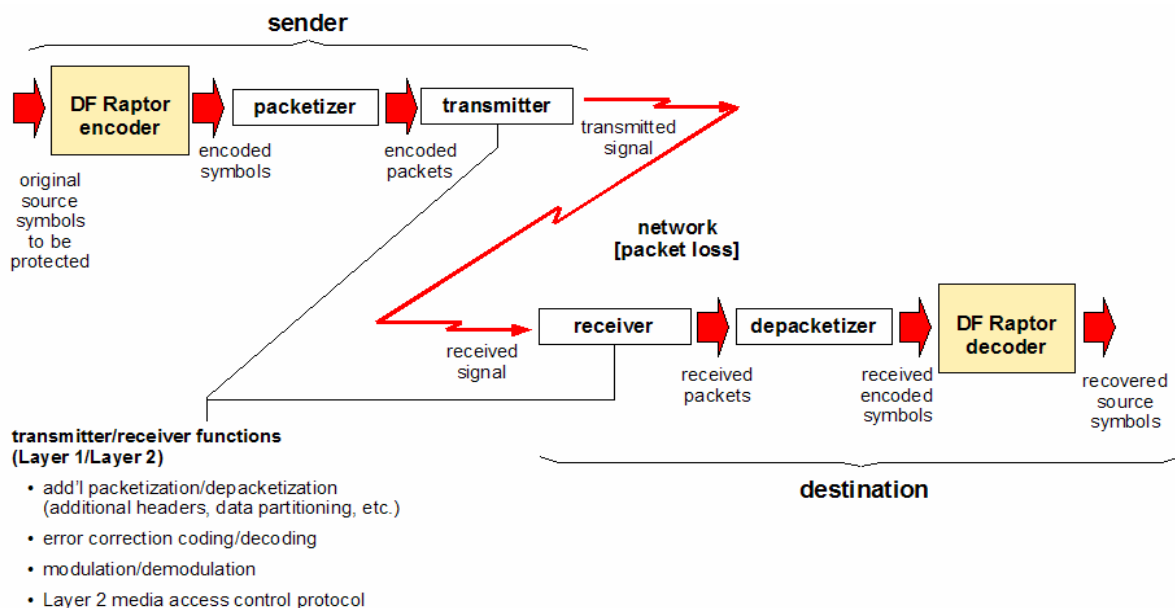
DF Raptor™ is the world's most advanced forward error correction (FEC) technology for data networks.

DF Raptor is an erasure correction code, capable of correcting “missing” or “lost” data. By recovering lost data packets without requiring retransmission from the sender, DF Raptor efficiently and effectively provides reliability in data networks. And, like a water fountain producing an endless supply of water drops, any of which can be used to completely fill a glass, DF Raptor is a fountain code (a “digital fountain”) that can generate an unlimited number of encoded output symbols, any of which can be used to recover the original input symbols. DF Raptor offers:

- The ability to generate a potentially limitless amount of encoded data from any original set of source data, thereby supporting extremes of low or high network loss as needed for a particular application
- The ability to recover the original data from an amount of successfully received encoded data only slightly greater than the original source data, regardless of which specific encoded data has been received
- Exceptionally fast encoding and decoding algorithms -- operating at nearly symmetric speeds that grow only linearly with the amount of source data to be processed and independently of the actual amount of loss – so that DF Raptor is highly scalable and suitable for lightweight software-based solutions

DF Raptor can be employed at the application or transport layer to provide reliability to data communications. Easily integrated into a specific use, DF Raptor optimally protects streaming media or data distribution applications from the effects of packet loss no matter what the source.

DF Raptor guarantees error-free delivery in end-to-end communications over any data network, enhancing the quality of streaming content or accelerating the delivery of data. At the same time, DF Raptor provides complete flexibility in balancing and optimizing a specific application's requirements for bandwidth utilization, latency, and degree of packet loss protection. And because the algorithms employed by DF Raptor can be implemented as software on relatively low-power platforms, DF Raptor enables a variety of new applications and services.



FEC Technology

In order to protect data transmitted from a sender to a receiver, FEC technologies involve encoding algorithms that add some degree of redundancy – additional repair data – to the original source data to form the encoded data. For a specific encoding algorithm, the complementary decoding algorithm allows the receiver to detect and possibly correct errors in the original data solely on the basis of the received encoded data. The error correction is “forward” in the sense that no feedback from the receiver to the sender or further transmission by the sender is required.

The additional redundancy introduced by FEC implies that more than just the original data is transmitted, resulting in either a longer transmission time (if the data rate is maintained constant) or a faster data rate and thus higher bandwidth occupancy (if the transmission time is maintained constant). Paradoxically, however, the additional redundancy can ultimately save transmission time and bandwidth use when compared to the retransmissions that would be needed without FEC. The fundamental trade-offs posed by FEC are the degree of error protection provided by a particular algorithm, how much encoding and decoding processing is required, how much latency is introduced, and how much additional overhead or bandwidth expansion is needed to protect against error or loss.

FEC techniques encompass error detection codes, error correction codes, and erasure correction codes, where each type of code serves a different purpose:

- Error detection codes allow the receiver to determine whether the received data is in error but do not provide the means to identify and correct the errors. For example, a 1’s complement checksum error detection code is used as part of IP data packets to allow the receiver to verify the integrity of the IP header and in TCP and UDP data packets to verify the integrity of the header and payload data.
- Error correction codes allow the receiver to identify and correct up to a certain number of errors occurring in the received data. For example, convolutional or block error correction codes are used in the physical layers of 802.11a/b/g Wi-Fi devices and DOCSIS cable modems to compensate for the bit error rates associated with those channels.
- Erasures correction codes allow the receiver to correct up to a certain amount of missing data where the positions of the missing data within the total amount of data are already known. For example, DF Raptor can be used in any packet data network to recover packets that have been “lost” either because they were not ever received or because errors were detected and the packets were discarded.

As an erasure correction code, DF Raptor can be used to provide packet-level protection at the transport layer or higher, augmenting the bit-level protection that may be provided by the link and physical layer protocols and their use of error detection or error correction codes.

Sources of Packet Loss

In packet-switched networks, a data file or a stream of data is divided into data packets that are individually transmitted and processed. Each packet is composed of its digital payload -- the data that is being conveyed -- and the header and trailer bits that are used by the network’s protocols to deliver the data to its destination. DF Raptor provides reliability, protecting against packet loss by ensuring that all the packets associated with a block of data are available error-free at the destination regardless of packet loss, congestion, or latency.

In a data network, packet loss generally occurs for two different reasons: 1) discarding of packets along the transmission path; or 2) data corruption such that any bit-level error correction code that might be used by the physical or link layer cannot restore the full packet and it is effectively lost. Sources of packet loss thus include:

- Instantaneously high network traffic levels can cause buffer overflows at intermediate routers, causing packets to be discarded en route to their ultimate destination
- Congestion control algorithms can result in intermediate routers deliberately discarding packets
- Network quality can be impaired; for example, specific links within the network can be momentarily disrupted by equipment faults, power failures, configuration errors, or denial of service attacks

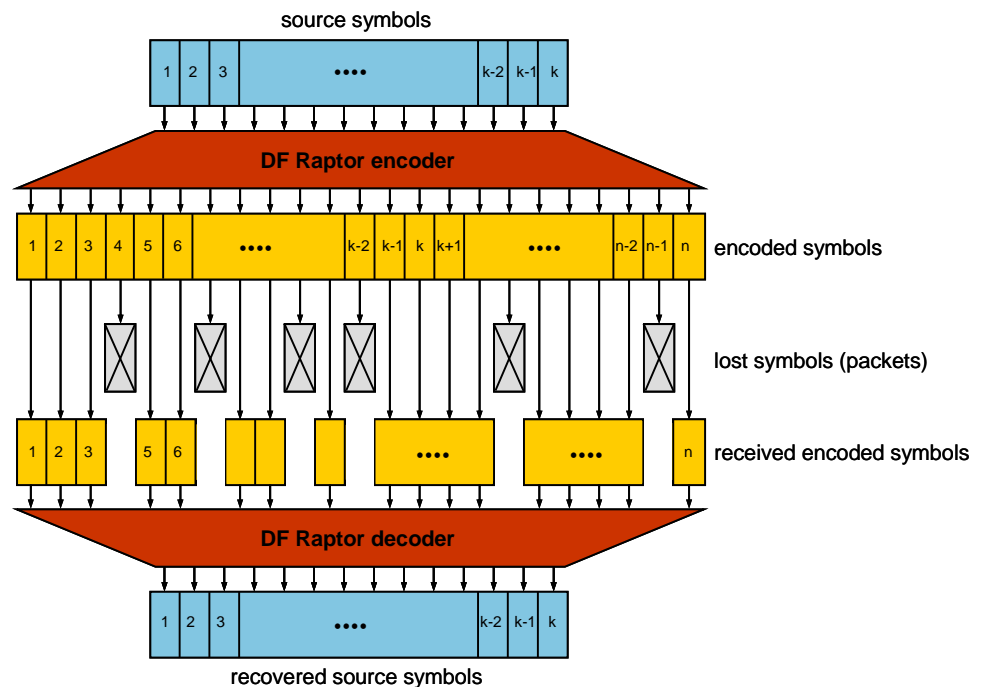
- Data quality as a result of impairments to the physical channel can cause a packet to be discarded at the destination. Different access and data distribution technologies are prone to different sources of impairments:
 - xDSL and cable connections can be affected by crosstalk, poor quality wiring and connectors, or local electro-magnetic interference
 - fiber optic connections can be susceptible to physical vibration, temperature variation, or poor quality splices and temporary connectors
 - broadband wireless access, cellular, and satellite links can be disrupted by radio frequency interference, signal attenuation due to line-of-sight obstructions, multipath, weather-induced fading, poor cabling, or antenna pointing, polarization, or alignment errors
 - on-premise data distribution can be of poor quality, especially if unmanaged wireless local or personal area network (WLAN or WPAN) technology is employed

DF Raptor Provides Protection Against Packet Loss

Digital Fountain's DF Raptor code has been designed and optimized for flexibility, low complexity, and powerful erasure protection. DF Raptor is a packet-level erasure code, protecting against the loss of packets that may occur in transmitting data across a packet-switched network.

The data to be encoded is first divided into fixed-size elements - source symbols – that are the atomic units upon which the DF Raptor algorithm operates. The source symbol size is chosen such that each source symbol will be mapped into a complete data packet or some fraction of a packet when it is transmitted.

The DF Raptor encoder then processes a block of source symbols – for example, a complete data file to be transmitted or a block of a continuous data stream – into a larger block of any desired length of encoded output symbols. Each encoded symbol is the same size as a source symbol and is determined by the bit-wise XOR of a number of source symbols according to the DF Raptor algorithm. Each encoded symbol is generated independently of the others, and, because DF Raptor is a fountain code, there is no limit to the number of encoded symbols that can be generated from a given source block. The generated output symbols are then packetized as necessary for transmission over the network.



DF Raptor encodes a block of k source symbols as $n > k$ encoded symbols in order to protect the source symbols from loss

At the receiver, not all the encoded symbols may be received because of lost or corrupted packets. As long as enough encoded symbols are successfully received, however, the DF Raptor decoder can recover the original source symbols. The number of encoded symbols that needs to be successfully received in order to fully recover the data is only slightly greater than the number of source symbols in the source block. It makes no difference to the DF Raptor decoding algorithm which specific encoded symbols are received or in what order they are received -- as long as an encoded symbol is received, it can be used to recover the original data.

How Many Encoded Symbols Are Necessary?

DF Raptor can generate as many encoded symbols from a source block as needed to counter the effects of packet loss. Alternatively, DF Raptor can generate as few encoded symbols as needed to limit the amount of bandwidth expansion or additional transmission time yet still provide a desired level of protection against packet loss. DF Raptor provides full flexibility -- the choice is up to the system designer and the requirements of the specific application.

As a fountain code, DF Raptor allows the possibility of system designs that adaptively adjust to protect against the actual amount of experienced packet loss. For example, DF Raptor-encoded symbols can be continually generated and transmitted until enough have been received to fully recover the original data, at which time the receiver can transmit an acknowledgement to the transmitter requesting that transmission cease or that the next source block be transmitted. With this approach, the overhead associated with FEC coding can be held to a minimum while still providing complete protection against any level of packet loss. In applications where mission-critical data must be received in its entirety without errors, DF Raptor provides a unique solution.

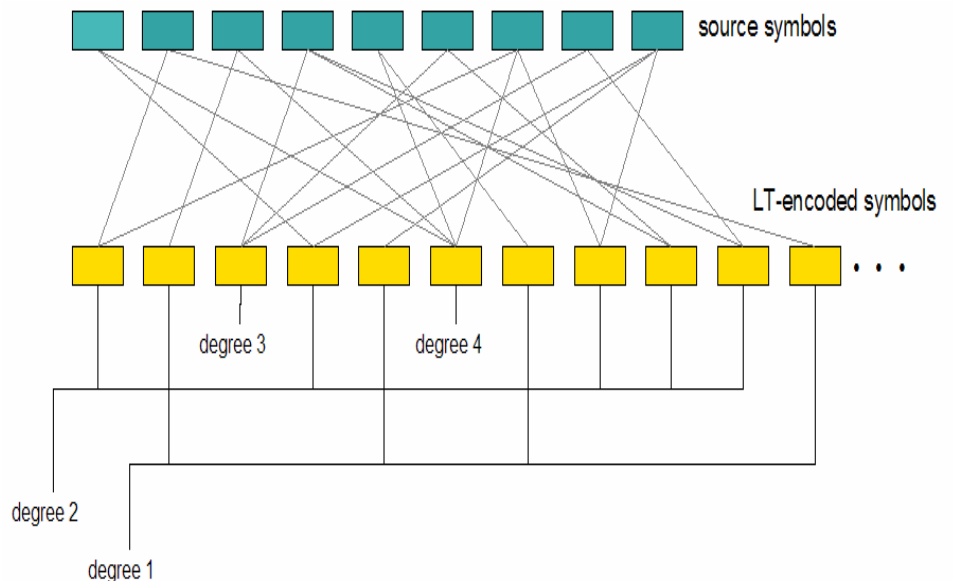
In many applications, however, the amount of DF Raptor-encoded data that can be used may be limited because of bandwidth or time constraints. For example, in streaming media applications, DF Raptor can be used to protect consecutive blocks of the data stream by encoding each block in turn and generating a new stream of encoded blocks of data. In that case, each encoded block must typically be transmitted over the same duration as occupied by the original source block, requiring transmission at a faster data rate than the original stream and expanding the occupied bandwidth. The allowed amount of bandwidth expansion then determines how many encoded symbols can be used and thus the level of protection against packet loss.

As another example, in many data broadcast applications there is no return channel available for all the receivers to acknowledge successful receipt of the entire data file. With DF Raptor, each received encoded symbol can be used to help recover the source data, thereby making it possible to limit the total transmission time based on the expected level of packet loss and the desired probability that all receivers obtain the data. Again, DF Raptor provides full flexibility for whatever the application may be.

The DF Raptor Encoding and Decoding Algorithms

DF Raptor is defined by encoding and decoding algorithms that were specifically created to produce a fountain code where the amount of processing required to encode or decode a block of data increases linearly with respect to the length of the block of source symbols.

The DF Raptor encoding process is simply an algorithm to construct new encoded symbols by computing the bit-wise XOR of specific source symbols. The encoding process can be seen as a graph with connections linking each encoded symbol to the source symbols that are XOR'ed to create it. The decoding process is then simply the reverse: given an adequate number of received encoded symbols and knowledge of the XORs used to create each one, solve for all the source symbols.



DF Raptor employs an LT fountain code to generate an unlimited number of encoded symbols from the source symbols

A core component of DF Raptor is an LT code. Each LT-encoded symbol is randomly and independently generated as the XOR of a particular subset of the source symbols. The degree of each LT-encoded symbol is the number of source symbols used to generate it, and the key to the LT algorithm is the use of a probability distribution to govern the degree of each encoded symbol.

Each LT-encoded symbol is generated according to the following algorithm:

- A specific degree d is chosen randomly according to the degree distribution
- Given the degree d , the specific d source symbols are chosen randomly according to a uniform distribution
- The encoded symbol value is then the bit-wise XOR sum of the selected source symbols

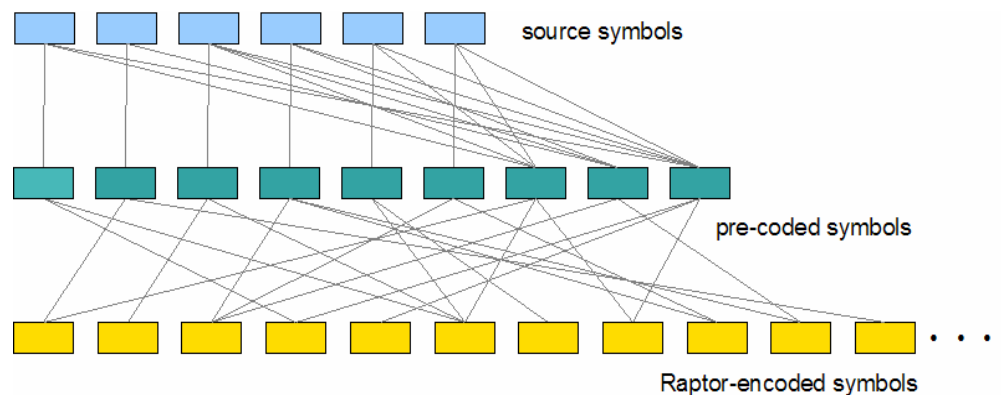
Each LT-encoded symbol is then transmitted along with information to the receiver indicating the degree and the selected source symbols (for example, the information can be the seeds used by a common pseudorandom number generator). With this knowledge, the decoder at the receiver knows which specific d source symbols were used to generate each encoded symbol, although the values of the source symbols are unknown until decoding is complete.

Because each encoded symbol is generated independently and at random using the same degree distribution, all received encoded symbols are equally valuable in recovering the source symbols. And because any number of encoded symbols can be produced simply by repeating the algorithm, an LT code is truly a fountain code.

The complexity of encoding and decoding an LT code is directly related to the degree distribution. The smaller the average degree is, the less the number of XORs involved in calculating each encoded symbol and the simpler the encoding and decoding processing. At the same time, the degree distribution must allow the decoding process to fully recover the entire source block with a number of received encoded symbols only slightly larger than the total number of source symbols. Well-performing degree distributions for LT codes have been determined, but the resulting complexity is not linear with respect to the number of source symbols.

In order to achieve linear complexity, DF Raptor encodes in two stages: first, a low-complexity pre-coding algorithm is applied to the input block of source symbols to create a pre-coded block, and then an LT code is applied on the pre-coded block to generate an unlimited number of encoded symbols from the pre-coded block. Decoding then involves the inverse operations: using the received encoded symbols to recover the pre-coded symbols, then using the recovered pre-coded symbols to recover the source block.

The LT code employed in DF Raptor uses a degree distribution that provides low complexity at the expense of loss protection. Even though the LT code may not allow full recovery of the pre-coded symbols at the decoder, the pre-coding algorithm can operate on whatever pre-coded symbols have been recovered in order to then fully recover the source block.



DF Raptor encoding uses first a pre-code algorithm and then an LT fountain code to achieve optimum performance

The performance of DF Raptor has been optimized by careful design of the pre-coding algorithm and of the degree distribution used in the LT code. The unique properties of DF Raptor are due to the analysis used to optimize it and are not simply due to its algorithms. As a result, DF Raptor has encoding and decoding complexity that is linear with the source block size and allows the source block to be fully recovered using a number of received encoded symbols just slightly greater than the total number of source symbols.

Processing Requirements

DF Raptor is typically implemented as software/firmware operating on a general-purpose processor without the need for dedicated hardware. Performance depends on many variables, especially processor and memory access speeds. Because DF Raptor is linear with respect to the source block size, however, the processing demands do not vary with the particular level of packet loss protection provided – any level of loss protection incurs the same degree of processing complexity. Maximum encoding and decoding speeds on a particular platform are generally quite high, where these speeds are especially important for streaming applications in which decoding and, for 2-way interactive applications, encoding times add to latency. Ultimately, the low processing requirements associated with DF Raptor imply low costs – often, processor utilization is such that no additional processing power is required to support DF Raptor.

DF Raptor thus makes possible inexpensive solutions for a wide range of applications. In particular, DF Raptor is well-suited for use in consumer-grade electronics such as set-top boxes, PDAs, and cellular phones, running on such a device's general-purpose processor without any supporting hardware. For example, at typical streaming rates, DF Raptor will use very little processing power – a 206 MHz Intel StrongARM processor is capable of decoding DF Raptor-encoded blocks at rates greater than 27 Mbps, much faster than would generally be needed in a real application. And, depending on the specific implementation, the code space required for DF Raptor is ~100-300 kB so that the cost implications of DF Raptor are minimal.