Network Working Group
Internet-Draft
Intended status: Experimental
Expires: February 21, 2013

J. Zinky
A. Caro
Raytheon BBN Technologies
G. Stein
Laboratory for
Telecommunications Sciences
August 20, 2012

Bundle Protocol Erasure Coding Extension
draft-irtf-dtnrg-zinky-erasure-coding-extension-00

Abstract

This document describes an extension to the Delay and Disruption Tolerant Networking (DTN) Bundle Protocol specification [RFC5050] which describes a protocol that enables the transfer of relatively large Data Objects over disrupted networks. The Erasure Coding Extension is a mechanism that extends the ability of the existing DTN bundle fragmentation mechanism to handle situations where bundles have a high probability of being dropped. An example use case is a situation where no communication contact period will ever be long enough to send the whole Data Object. In this case the object must be partitioned into smaller chunks and these chunks are sent in multiple bundles. The Erasure Coding Extension provides a recovery mechanism that allows many of these bundles to be dropped and still allow the whole Data Object to be successfully sent.

This document describes an Erasure Coding Extension Block and a framework for integrating Forward Error Correction (FEC) into the bundle protocol. The Erasure Coding Extension is designed to support multiple FEC schemes and content object types. This is the framework document for a series of documents about erasure coding for DTN. Companion documents describe specific FEC schemes [RandBinary] and specific Data Object types [EcObjects].

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 21, 2013.

Copyright Notice

Table of Contents

1. Introduction

Delay and Disruption Tolerant Networks (DTNs) [RFC4838] have extreme communication constraints. DTNs can include high link delays, such as hours to reach another planet, or no guarantee of a contemporaneous end-to-end path between source-destination pairs, such as a bus with a DTN Bundle Protocol Agent ferrying data to a remote village. These constraints limit the timeliness of feedback that can be sent from the receiver back to the sender to acknowledge receipt or to control the data flow. Forward Error Correction (FEC) techniques are a potentially important mechanism for DTN networks, especially if a large Data Object must be partitioned into multiple bundles [Erasure_Wang]. While the base Bundle Protocol [RFC5050] has a fragmentation feature to partition a large bundle into multiple smaller bundles, there is no support for FEC, i.e. all bundle fragments must be received to reconstruct the original bundle.

To correct for this lack, this document describes an extension to the Bundle Protocol to support FEC mechanisms. The extension allows for large Data Objects to be partitioned into smaller Chunks. A FEC coding scheme is used to encode the Chunks into Encoding Bundles. Redundant Encoding Bundles are transmitted to allow recovery of Bundles that where dropped. The specification allows for different tradeoffs in the level of protection, overhead, and computation needed to encode and decode large Data Objects, based on the kind of FEC coding scheme, the network characteristics, and the type of Data Object.

This document describes a FEC framework for the encoding, decoding, forwarding, and recoding within a DTN network. A Bundle Protocol Extension Block is defined to send the parameters needed for several FEC coding schemes. The Erasure Coding extension is flexible and supports multiple FEC coding schemes and Data Object types. Separate documents define how specific FEC coding schemes [RandBinary] and Data Object formats [EcObjects] support this extension. Future documents will describe protocols for erasure coding flow control, routing, and recoding.

This document assumes familiarity with Forward Error Control techniques and the FEC framework described in "Forward Error Correction (FEC) Building Blocks" [RFC5052], this paper uses the Bundle Protocol as a "Content Delivery Protocol" that will use FEC schemes to insure reliable delivery of Data Objects. Many FEC coding schemes MAY be supported, such as Random Binary Codes [RandBinary], block-oriented parity [RFC5445], Raptor Codes [RFC5053], Reed-Solomon [RFC5510], and Low Density Parity Check (LDPC) [RFC5170]. The exact coding scheme used depends on the DTN network environment, as no one coding scheme works optimally in all situations. The Erasure Coding

extension could also support Network Coding techniques.

2. Terminology

The following definitions describe terms used in the Bundle Protocol Erasure Coding Extension. The terminology for Delay and Disruption Tolerant Networks (DTNs) follows [RFC4838],[RFC5050], and Forward Error Coding (FEC) terminology follows [RFC5052].

2.1. Definitions

Terms are grouped by a common domain. Terms within a domain have relationships with other terms in that domain and not with other domains.

2.1.1. Delay and Disruption Tolerant Network Terms

Delay and Disruption Tolerant Networks (DTNs) use the bundle protocol suite [RFC5050] to send data over networks with extreme constraints, such as high-delay space channels with propagation delays in hours or networks with no contemporaneous end-to-end path.

Bundle is a DTN protocol data unit (PDU) as defined in [RFC5050].

Bundle Node is any entity that can send and/or receive bundles as defined in [RFC5050]. In the context of this document, a Bundle Node may be either a DTN Application or a Bundle Protocol Agent.

Bundle Protocol Agent (BPA) offers DTN services and executes the procedures of the bundle protocol. A BPA performs a store and forward function, receives, processes, and sends bundles as defined in [RFC5050]. A BPA may have any Erasure Coding process, e.g. Encoder, Decoder, Intermediary Regulator or an Intermediary Recoder.

DTN Application generates and receives bundles in order to create an end-user application. A DTN application may have only an Erasure Coding Encoder or Decoder process.

2.1.2. Forward Error Correcting Terms

Data Object is an ordered sequence of octets that is transferred as a single unit as defined in the FEC Framework [RFC5052]. The object has a defined format, such as a file, a large bundle, a stream, or a mime type. A Data Object has a UUID, which marks all Encodings that belong to the same Data Object.

Chunks are ordered pieces of a Data Object before it is encoded. All Chunks are the same length with the last Chunk being padded when necessary. A Chunk is called a Source Symbol in the FEC Framework[RFC5052] .

Encoding Vector is the coefficients for the coding formula used to create an Encoding. An Encoding Vector carries the FEC Payload ID as defined in the FEC framework [RFC5052].

Encoding Data is the result of applying a coding formula to the Chunks of a Data Object. The Encoding Data is the same length as the Chunks. An Encoding Data may carry either a Source Symbol or a Repair Symbol as defined in the FEC framework [RFC5052].

Encoding contains the corresponding Encoding Vector and Encoding Data, with a Transfer Specification.

Encoding Set is a group of Encodings that all belong to the same Data Object (same UUID). The Encodings in an Encoding Set represent a set of linear equations. When the rank of the Encoding Set is equal to the number of Chunks in the Data Object, the Encoding Set linear equations can be solved to decode ALL of the Chunks for the Object.

Innovative Encoding: When an Encoding is added to an Encoding Set, the Encoding is said to be "Innovative" relative to the Encoding Set, if the new Encoding adds new information to the Encoding Set (i.e., increases the Encoding Set's rank).

Redundant Encoding: When an Encoding is added to an Encoding Set, the Encoding is said to be "Redundant" relative to the Encoding Set if the new Encoding does not add new information to the Encoding Set (i.e., the Encoding Set's rank stays the same).

Duplicate Encoding: Two Encodings are equivalent or duplicate, if they belong to the same Data Object (same UUID) and have the same Encoding Vector and hence the same Encoding Data.

2.1.3. Erasure Coding Protocol Terms

Encoding Bundle contains the information necessary to send an Encoding using the Bundle Protocol. Some information is put in the bundle primary block, such as the Data Object's Transfer Specification. Some information is put in the Erasure Coding Extension Block, such as the Encoding Vector. Some information is put in the Bundle payload, such as the Encoding Data.

Erasure Coding Extension Block exposes Encoding information that is needed by Intermediate Regulators. Specifically, it includes the Encoding Vector and Handling Specification.

Erasure Payload stores the part of an Encoding that is not exposed to Intermediate Regulators. Specifically, it includes the Encoding Data.

Handling Specification defines the importance of forwarding Encoding Bundles relative to its Encoding Set or other Encoding Sets. The Handling Specification MAY change the order, priority, or rate for sending Encoding Bundles.

2.2. Abbreviations

BPA: Bundle Protocol Agent, see [RFC5050]

DTN: Delay and Disruption Tolerant Network, see [RFC5050]

EID: End-point Identifier, see [RFC5050]

FEC: Forward Error Correction, see [RFC5052]

UUID: Universally Unique Identifier, see [RFC4122]

SDNV: Self-Delimiting Numeric Values, see [RFC6256]

2.3. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Erasure Coding Architecture

The goal of the Erasure Coding extension is to enable the transfer of large Data Objects over disrupted links. The Erasure Coding Architecture defines protocol layers, protocol functions, and components that achieve this goal in a modular and extensible manner. Besides defining terms and relationships, when realized the architecture forms a distributed system that executes at multiple locations, cross cuts multiple protocol layers, and reuses services from external components in a DTN network.

Erasure Coding happens at multiple locations across a DTN. At the end points, Data Objects are divided into Chunks, which are Encoded at the sender and Decoded at the receiver. Along the way between the sender and the receiver Intermediate Regulator, Intermediate Recoders and Bundle Forwarders may process the Encoding Bundles to ensure the end-to-end delivery of a Data Object, despite the poor communication channel between the sender and receiver.

This section describes the end-to-end Erasure Coding process, including the functions that need to be performed and the layers at which they are performed. This section also describes several configuration use cases and how the components carry out their roles. Subsequent sections describe each of the protocol layers: Data Object Content Layer, the Coding Layer, the Intermediate Regulating Layer, and the Bundle Forwarding Layer.

3.1. Erasure Coding Process

A Data Object goes through a series of transformations as it is transmitted using the Erasure Coding protocol: from a Data Object to Encodings to Bundles and back again from Bundles, to Encodings, to a copy of the Data Object at the receiver. The transmission process has the following steps:

Step 1: A Data Object has content that needs to be transferred across a DTN as a single whole. If any part of the content is missing the Data Object is not considered transferred.

Step 2: The Data Object's list of Octets is divided into Chunks of equal length. The last Chunk MUST be padded so that it is the same length as the other Chunks. The order of the Chunks MUST be preserved to retain the ordering of the Data Object octets. The ordered list of Chunks that represent the Data Object is given an unique identifier (UUID).

Step 3: Chunks are coded into an Encoding. Some Combination of Chucks are combined using one of serval possible FEC schemes. The coefficients of the coding formula are stored in an Encoding Vector. The results of the combination are stored in an Encoding Data, which is an ordered list of octets that is the same length as a Chunk. Together the Encoding Data, Encoding Vector, the coding scheme, and the Data Object UUID form an Encoding. All Bundle FEC coding schemes MUST create Encodings with this structure.

Step 4: All the Encodings generated for a Data Object share the same UUID and may be grouped into an Encoding Set. A Full Encoding Set has enough Encodings to extract the Data Object. Note that an Encoding Set may have Redundant Encodings so that not all the Encodings in the set need to be received. The level of redundancy is determined by the FEC scheme, the application Quality of Service requirements, and the expected transmission characteristic across the DTN.

Step 5: Each Encoding is placed in an Encoding Bundle. The Encoding Vector, UUID, and FEC Scheme are formatted into the Erasure Coding Extension Block defined in this document. The Encoding Data is formatted into the Bundle Payload.

Step 6: Transfer specification is added to the Encoding Bundle to help resolve trade offs in the transfer of this Encoding Bundle relative to other bundles. The transfer specification cross cuts multiple layers by adding parameters to the bundle header, extension blocks, and Data Object's internal metadata.

Step 7: When Encoding Bundles arrive at a destination, they are added to a Encoding Set based on their UUID. Duplicate Encodings have the same Encoding Vector as an Encoding already in the set and should be dropped. Redundant Encodings do not add any new information to the Encoding Set and may be dropped by the Decoder, but may be used by Recoders.

Step 8: The Data Object is reconstructed at a destination as an array of octets from the decoded Chunks. When an Encoding Set has enough Encodings to decode some of the Chunks. The Chunks are extracted and sent to the application. Some FEC schemes may decode some Chunks as Encodings arrive, but other FEC schemes may not be able to decode any Chunks until nearly all Encodings have arrived. For some FEC schemes, the decode operation may consume substantial cpu and memory resources.

3.2. Erasure Coding Functions

The Erasure Coding process performs several functions that happen at different protocol layers and may happen at different locations within the DTN network.

Encoder converts a Data Object into Encodings Bundles and sends the bundles into the DTN. The Encoder is responsible for converting the Data Object into Chunks, coding the Chunks into Encodings, packing the Encoding into a Bundle, and setting the Bundle header parameters, to meet the transfer specification requirement. The Encoder MUST send enough Encodings to allow a Decoder to reconstruct the Data Object.

Decoder receives Encodings Bundles, extracts Encodings, and stores the Encodings into an Encoding Set. The Decoder solves the coding equations to reconstruct Chunks for the Data Object. When enough Encodings arrive to solve the Encoding Set, the resulting Chunks are given to the consumer of the Data Object. Note, some FEC schemes may allow solving for some Chunks before solving for all Chunks. In this case, the Chunks MAY be delivered as they are solved or as a complete Data Object.

Intermediate Regulator is a process on the path between the Encoder and Decoder. It may change the order of Encoding Bundles to give one Data Object priority over other Data Objects. It may manipulate how Encoding Bundles are sent between BPAs, based on fields in the Encoding Extension Block, such as UUID, handling spec, and Encoding Vector. Also, it may round-robin through Encoding Bundles in a Encoding Set to increase the diversity of Encodings being forwarded during different contacts. Some forwarding rules may depend on not sending Redundant Encodings to neighbors that have been met on previous contacts.

Intermediate Recoder is a source of Redundant Encodings that may be manipulated and forwarded by the Intermediate Regulator. The Intermediate Recoder creates new Encodings from an Encoding Set held within the Intermediate Node. The new Encodings are redundant, but not duplicate to the other Encodings in the Encoding Set. The Intermediate Regulator MAY send these Redundant Encodings across different paths to the destination. Note that the Intermediate Recoder does not have to solve the Encoding Set. A generated Encoding is created by just combining multiple Encodings from the Encoding Set.

Bundle Forwarder is a legacy DTN BPA that does not understand the Erasure Coding Extension. It routes and forwards Bundles based on fields in the standard Bundle primary block header.

These functions are grouped into four layers. The Data Object Content layer is the external consumer of the DTN services and is concerned with transferring a Data Object from the source to the destination(s). The Coding Layer encodes and decodes the Data Object into Encodings Bundles. The Intermediate Regulating Layer regulates the Encoding Bundle flows across the DTN. The Bundle Forwarding Layer routes and forwards the Bundles along the path from the source to the destination(s).

3.3. Erasure Coding Component Configurations

Erasure Coding architectural components are locations that execute the Erasure Coding functions. The architectural components are contained inside either DTN Applications or DTN Bundle Protocol Agents (BPAs). The end-to-end Encoders and Decoders may be located in either DTN Applications or DTN BPAs. Intermediate Regulators and Intermediate Recoders may only be located in DTN BPAs. The links between components are either through internal interfaces or across a DTN.

Different configurations of the Erasure Coding Components allow for the creation of distributed systems that meet different end-user application requirements, within the constraints of the DTN. These configurations help integrate legacy DTN applications and BPAs with Erasure Coding Components, selectively adding or removing functionality depending on availability.

Erasure Coding-aware DTN Applications preform the Encoding and Decoding functions with in the application itself. The application has knowledge of the end-to-end transfer specification and sets the fields in the Bundle primary block and Handling Specification to best meet the requirements. Also, if the FEC scheme allows for decoding some Chunks before all Chunks are ready, these early Chunks may be consumed by the application.

Legacy DTN Applications that send large bundles, the Encoding functions may be located in the BPA. In this case, a large Bundle is encapsulated as a Data Object and sent to the destination BPA where the large Bundle is decoded and reinserted into the DTN routing layer.

Erasure Coding-aware BPAs The use of Intermediate Regulators and Intermediate Recoders enables traffic shaping based on the UUID, detection of Redundant Encodings, and changing the order of Bundle transmission. The use of intermediate Recoders allows sending more diverse Encodings, substantially reducing the number of Encoding Bundles that need to be received before the Encoding Set can be solved for a Data Object.

Legacy BPAs are not aware of the Erasure Coding Extension, but they may still forward bundles. No Erasure Coding specific processing may be done along the path to improve the importance and urgency of the transfer. Only the existing Bundle-layer Quality of Service mechanisms may be used, such as lifetime, priority, and duplicate detection.

All combinations are possible including running Erasure Coding-aware Applications over Erasure Coding Aware BPAs with Encoders and Decoders. In this case, the BPA MUST check for the Erasure Encoding Extension Block on all Bundles inserted by the application. The BPA MUST NOT attempt to encode an Bundle that has already been encoded by the application.

4. Data Object Content Layer

The goal of the Data Object Content Layer is to transfer a Data Object from a source to a destination within some application specific Quality of Service requirements. It uses the DTN Bundle protocol with Erasure Coding Extension to accomplish the transfer.

The Data Object Content Layer is concerned with Data Objects and a Transfer Specification for that Data Object. The Erasure Coding Extension expects to transfer an ordered array of octets across a DTN. The job of the Data Object layer is to map the Data Object and its Transfer Specification into the mechanisms available from the Erasure Coding Extension. This mapping is complex and very application and Data Object type specific, but mappings share some common features described in this section.

The Data Object type defines the internal representation of a Data Object's content and meta data as an ordered array of bytes. Many Data Object types are supported. A companion document [EcObjects] defines the Data Object types for files and large bundles.

The Data Object layer has to interact with all three other layers.

For the Bundle Forwarding Layer, the Data Object layer MUST set Bundle service delivery parameters, such as lifetime and priority, so that all Encoding Bundles for a Data Object have the same value. We do not recommend requesting delivery notifications, especially custody notification as these administrative services generate feedback traffic at the Bundle Layer and not at the Coding Layer. The destination is specified in the form of an DTN EID. The transfer specification may require multiple destinations, which may be satisfied by some configurations of the DTN BPAs routing protocols. For example epidemic routing attempts to flood Bundles to all nodes in the DTN or alternatively Geo-role routing forwards Bundles to all nodes playing a role within a geographic region. The Bundle Destination EID is used to specify the routing protocol and its parameters. Features from other DTN extensions may be used by the Data Object Layer. For example, the Bundle Security Protocol MAY be used to protect the authenticity of the Bundle extension blocks and payload.

For the Coding Layer, the Data Object and its meta data MUST be converted into an ordered array of octets. The Transfer Specification MAY have meta data that needs to be sent end-to-end with the Data Object. The representation of the meta data depends on the Data Object type. Likewise, some Data Object types may want control over how Chunks are made and whether Chunks are delivered early. For example, a video Data Object may want to align Chunks on

video frame boundaries and may want to have the key frames delivered early. These Data Object format issues are addressed by companion documents for each Data Object Type, such as [EcObjects]

For the Intermediate Regulating Layer, the Data Object Layer SHOULD set parameter values to the Handling Specification. Intermediate Regulators MAY use the parameters to tradeoff resource consumption and delivered service among competing Data Objects.

4.1. Data Object Transfer Specification

The transfer specification is an abstract representation for _how_ a Data Object should be transferred from the source to the destination. It gives hints to the Erasure Coding architectural components on how to create, combine, transfer, and decode Encoding Bundles. The transfer specification takes end-to-end quality of service requirements and translates them into the Bundle Protocol mechanisms necessary to meet those requirements. A transfer specification MAY be manifested physically in an Erasure Coding implementation, but its main role is to illustrate the control points available at each protocol layer and how they interact.

The transfer specification is not monolithic. It cross cuts different protocol layers and its content is spread among different fields in the bundle header, extension block, and the bundle payload. At the bundle layer, it specifies how Encoding Bundles are handled relative to other DTN traffic. At the Regulating layer, it specifies how Encoding Bundles are handled relative to other Encodings Bundles for the same Data Object UUID or relative to other Data Object UUIDs. Finally at the Data Object layer, it specifies the properties of a copy of the Data Object at the destination.

In the current state of the Bundle Protocol, the BPA has no means for advertising its constraints to the Erasure Coding components and the components can not make active measurements of these constraints because the poor quality of the communication channel makes any measurements obsolete before enough samples can be collected. The process of choosing transfer parameters, such as Chunk Size, is thus an open loop process and depends on external oracles to predict the transfer constraints. As the Bundle Protocol evolves and adds capabilities for detecting and advertising its environmental constraints, the transfer specification will become more powerful and more formal. For example, the Bundle Security Protocol [RFC6257] should be used to meet the Erasure Coding requirement to authenticate the creator of Encoding Bundles. The functionality should be reused by the Erasure Coding extension and their is no need to redefine it here.

The following table lists some transfer specification parameters that are actionable in the context of the current Bundle Protocol [RFC5050] and the Erasure Coding extension.

Parameter	Type	Header	Field	Description
Destination	EID	Bundle	Destination	DTN endpoint destination(s) for Data Object. The endpoint may be a singleton or have multiple destinations.
Class of Service	Flags	Bundle	Priority	Priority relative to other Data Objects
Life Time	SDNV	Bundle	Life Time	Time when the Data Object should be deleted, if not delivered
Report Progress	Flags	Bundle	Processing Control	Quality of Service feedback for Data Object
UUID	Id	Extension	Unique Id	A Universally Unique Identifier that is associated with the Data Object transfer.

Number of Chunks	SDNV	Extension	Number of Chunks	The number of Chunks that the Data Object was divided into. Fewer Chunks reduce the reassembly time at destination.
Chunk Length	SDNV	Payload	Length	The size of the Chunks that the Data Object was divided into. Smaller Chunks reduce chance of bundle-layer fragmentation.
Handling Spec	Flags	Extension	Handling Spec	Handling hints for Intermediate Regulators for this bundle relative to other Encoding Bundles with same Data Object UUID
Authentication	ID	Bundle Security Extension	Signing	The level of trust in the Encoding components.
Data Object Metadata	Flags	Payload	Data Object Header in first Chunk	Meta Data about Data Object's systemic properties, such as authenticity, validity, and permissions

Table 1: Transfer Specification

4.2. Creating Chunks

The Data Object Layer formats the Data Object and meta data as an octet array that is used as input to the Coding Layer. The Data Object octet array is divided into an ordered array of equal length Chunks. Chunks are identified with an index. The Chunk with the index of '0' contains the octet with index '0'.

The Data Object Layer must add padding octets to the last Chunk, so that all Chunks are the same length. The Data Object Layer is responsable for storing the exact length of the Data Object without padding, because the Coding Layer does not have a field for the Data Object Length.

The exact number_of_chunks is determined at transfer time and is based on the Data Object Layer Transfer Specification and path characteristics of the DTN network. Decoders SHALL handle any number_of_chunks, but practical limits MAY be in the 1000's of Chunks. The chunk_length SHOULD be a multiple of 8. This will allow efficient octet array operations to be performed when encoding and decoding Chunks and Encoding Data.

5. Coding Layer

The goal of the Coding Layer is to divide a Data Object that is formated as an ordered array of octets into a group of Encodings. The Encodings MUST form a full Encoding Set and MAY have Redundant Encodings. The Erasure Coding Process (Section 3.1) is followed to encode Data Objects into Encodings and to decode an Encoding Set back into a Data Object. In this process, a FEC scheme is applied to the Chunks of a Data Object to form an Encoding. The FEC scheme uses a coding formula to convert the Chunks to an Encoding Data. The coefficients for the coding formula are stored in the Encoding Vector. The Encoding Vector and Encoding Data together form the Encoding. The exact format of the Encoding Vector and Encoding Data is defined by the FEC scheme. The Erasure Coding Extension supports multiple FEC schemes, such as the Random Binary FEC scheme defined in the companion document [RandBinary].

An Encoding is packed into an Encoding Bundle. In order for Intermediate Regulators to detect Redundant Encodings and group Encoding Bundles into Encodings Sets, some Encoding parameters are exposed in the Erasure Coding Extension Block (Section 5.1), such as the Encoding Vector and Data Object UUID. The Encoding Data is not needed by Intermediate Regulators and is put into the Bundle Payload. Intermediate Recoders need access to the content of both the Erasure Coding Extension Block and the Bundle Payload in order to generate new Encodings.

5.1. Erasure Coding Extension Block

The Erasure Coding Extension Block marks the Bundle containing an Encoding. The format of the Erasure Coding Extension Block is as follows:

Field	Type	Description
Block Type	Octet=0xEC	Bundle Protocol Extension Block Type is the constant 0xEC.
Block Processing Flags	SDNV	See Section 4.3 of [RFC5050] for bit settings
Block Length	SDNV	Length of extension block data, not including the padding.
Version	SDNV	Erasure Coding Extension Block version that increments with newer versions
Data Object Format Type	SDNV	Type number for the format of Data Object. Defines format for decoded Chunks, see [EcObjects].
Data Object UUID	UUID (128bits)	Universally Unique ID for a specific Data Object transfer. The UUID SHOULD be in the format specified in [RFC4122]. The Data Object Format specification MAY define a hash function to map the Data Object's name to a UUID. [NOTE: The reference implementation uses two SVDN fields to pack the upper and lower 64 bits.]
Handling Specification Length	SDNV	Number of SDNV Parameters in the Handling Specification. As the Erasure Protocol matures, parameters will be added to the Handling Specification. Version 1 of the Erasure Coding Extension does not define any Handling Specification parameters, so this field value is 0. Version 1 implementations MUST treat this field as a length of the next field, even though the content SHOULD be ignored.

Handling Specification Parameters	SDNV List	Hints on how Intermediate Regulators should order, prioritize, limit rate, or drop this Encoding relative to other Encoding bundles with the same Data Object UUID. No Handling Specification Parameters are defined for Version 1, so this field is null.
Number of Chunks	SDNV	Number of Chunks that the Data Object was divided into.
FEC Scheme Type	SDNV	The type number of the FEC Scheme used to interpret Coding Scheme Parameters, for example see [RandBinary].
FEC Scheme Parameters	Octets	Format of the octet array is determined by FEC Scheme. This field ends at the Block Length of the extension block data.
Padding	Octets	Extra octets so that the total length the whole extension block is a multiple of 4 octets.

Table 2: Erasure Coding Extension Block

The Data Object UUID field identifies the Data Object for the Encoding Bundle. The UUID MUST be unique in the DTN network over the life time of the bundle. While the UUID is treated just as a number, the UUID SHOULD be in the format specified in [RFC4122]. All the received Encoding Bundles with the same UUID form an Encoding Set for a Data Object.

The Data Object Format Type specifies the format used at the Data Object Layer. A Data Object format MUST define the headers and padding for the decoded Data Object. A Data Object format MAY define a format for the bundle payload, but by default the bundle payload contains just the Encoding Data. Several Data Object formats are defined in [EcObjects] and future documents.

The Handling Specification gives hints on how Intermediate Coders SHOULD process this bundle relative to other bundles in the same Encoding Set. For example, how many Redundant Encoding Bundles should be forwarded or what order to send Encoding Bundles across multiple contact points. The values of Handling Specification depend on both

the type of FEC scheme and the characteristics of the expected DTN communication path. The parameters (flags) of the Handling Specification MUST be actionable by Intermediate Regulators. For Version #1 of the Erasure Coding Extension Block, this handling parameters are undefined and Handling Specification Length MUST be zero (0).

The Encoding Vector is part of the extension block because Intermediate Regulators MAY want to determine the rank of an Encoding Set to detect Redundant Encodings. Calculating the rank is a less computationally expensive operation than decoding the Encoding Set for its Chunks. Calculating the rank only needs the Encoding Vector sand not the Encoding Data. Decoding needs to XOR multiple Encoding Datas together to decode each Chunk, a computationally expensive operation. The format of the Encoding Vector depends on the type of FEC scheme used. The common field among all FEC schemes is the Number of Chunks in the Data Object. The other two fields specify the type of FEC scheme and the bytes for the Encoding Vector itself. Several FEC schemes are defined in [RandBinary] and future documents.

More than one Erasure Coding Extension Block MAY be present in the same bundle. The interpretation of multiple extension blocks is to treat them as a composite formula that are merging Chunks from multiple Data Objects. The merged Encoding Vector adds the coefficients from the component Encoding Vectors. The merged Encoding Data (bundle payload) is the XOR of the composite Encoding Data. Intermediate Regulators MAY use the handling specification from either extension block to determine the handling procedures. The Handling Specification SHOULD be equivalent for all Erasure Coding Extension blocks in the bundle. The multiple extension block option supports experimentation in Network Coding.

6. Intermediate Regulating Layer

The goal of the Intermediate Regulating layer is to control the flow of Encoding Bundles as they move from the source to the destination. Intermediate Regulators decide on which Encoding Bundles should be sent next during a contact period with a neighbor BPA. Given the dynamic and complex nature of DTN topologies and the tradeoffs between competing DTN traffic flows, the traditional first come first serve queueing discipline is rarely adequate. Also, in the most general case the traffic shaping function in the Intermediate Regulators needs to work without feedback from neighbors or the end-to-end destinations.

Note: No reference implementation of the Intermediate Regulator have been completed at the time of this document publication. More research is necessary to determine the functionality of the Handling Specification and the policies used by Intermediate Regulators. This section is a place holder and lays out the issues. A future version of this section will capture the conclusions of the future research results.

6.1. Traffic Shaping

Traffic Shaping MAY improve the efficiency of resource consumption and the fairness between DTN traffic flows in the case where feedback is impractical between neighbors or between the destination and the source. Traffic shaping may determine when to stop forwarding Encoding Bundles from a Data Object, limit the rate, redundancy, or the order between bundles. The traffic shaping parameters may be calculated per Data Object UUID, per neighbor, or per contact.

A basic no feedback traffic shaper MAY be based on the following policy and parameters. During a contact the candidate Data Objects are ordered by importance (priority field in Bundle Primary Block) and urgency (lifetime in Bundle Primary Block). The highest priority with the earliest expiration is sent first. Sending a Data Object is divided into two phases; an Innovative Send Phase and a Redundant Send Phase. The Innovative Send phase sends only enough Encodings to form a full Encoding Set at the Receiver, while the Redundant Send Phase sends extra Redundant Encodings for FEC purposes.

Each phase has a limit on the number of Encodings in that phase and a limit on the rate for which Encodings are sent during that phase. Encoding Bundles are sent in Data Object order, sending all the Innovative Encoding Bundles for _all_ the Data Objects and then starting the redundant phase. While sending a Data Object during a phase, the next Data Object will start when the Max Number of Encodings for that phase is exceeded or the rate is exceeded. When

all the Redundant Encoding Bundles have been sent for all Data Objects, transmission stops.

The following are basic traffic shaping parameters that MAY be put into the Handling Specification. Version 1 of the Handling Specification does not allow these parameters to be specified dynamically, but defines a default Handling Specification based on these parameters.

Encoding Order defines the order to send Encodings for the same Data Object. The order may be changed based on the requirements of the transmission specification. The order MAY be Oldest First, Oldest Last, Round-Robin, or Random.

Send State defines the level of detail for sending a Data Object across contacts. For example, the encoding order MAY be be maintained for each time a Data Object is sent to a unique neighbor, or for each contact regardless of the neighbor, or no send state is maintained at all (Random).

Max Number of Innovative Encodings defines the number of Encodings in the Innovative Phase. For example, this limit MAY be greater than the Number of Chunks, if some Redundant Encodings should be included in the Innovative Send Phase. Conversely, the limit MAY be lower, if the contact periods are very short and Data Objects are expected to be sent over multiple contacts.

Max Number of Redundant Encodings defines the number of Encoding Bundles in the Redundant Phase.

Rate Limit for Innovative Encoding defines a limit on how quickly Encodings are sent rate limit during the Innovative phase. The rate MAY be specified using leaky bucket parameters, such as input period and initial fullness. Alternatively, a processor sharing model parameter MAY limit the maximum number Encodings to send before moving to the next Data Object with the same priority.

Rate Limit for Redundant Encodings defines a rate limit during the Redundant phase.

6.2. Handling Specification

The Handling Specification is a field in the Erasure Coding Extension Block that defines the parameters for traffic shaping and feedback messages. The format of this field is that of a length followed by a list of parameters with 'length' entries. The length and parameters are SDNV values, making the entire Handling Specification field have variable length. As Version #1 of the Erasure Coding Extension does

not define any Handling Specification Parameters, the length MUST currently be zero (0) for Version #1 and the parameter list null. As the Erasure Coding Extension matures, parameters will be added to the Handling Specification. The Handling Specification Length has the dual role of determining the number of parameters and as version indicator for the meaning of each parameter.

When the Handling Specification Length is zero (0), then the Intermediate Regulator and Recoder MAY use the following default policy. The order of sending Data Objects is based on the Priority and Lifetime fields in the Bundle Primary Block, with the highest priority first, then the earliest expiration. The order of sending Encodings within a specific Data Object is Round-Robin between contacts. The send state for a Data Object is saved for the whole Data Object regardless of neighbors. The maximum number of Encodings in the Innovative Phase is equal to the Number of Chunks. The maximum number of Encodings in the Redundant Phase should be the maximum of 10 and the square root of the Number of Chunks. There is no rate limit for sending Innovative Encodings or Redundant Encodings.

6.3. Feedback Messages

Feedback messages MAY be defined to increase the efficiency of resource usage for the case where feedback is possible between neighbors or between the destination and the source. The following types of feedback messages would enhance the exchange protocol.

Stop or End-to-End Acknowledgement messages that are sent back from the Destination Decoder to the Source Encoder to indicate that the Data Object has been received. The Source Encoder may use this acknowledgement to stop sending Redundant Encodings. Also, this could prompt the sending of a PURGE message, that is signed by the Encoder. The Stop message MUST expire no later than the original Data Object.

Purge messages to indicate that intermediate routers may remove all Encoding Bundles for a Data Object UUID. Encoding Bundles are valid until the Lifetime of the Bundle expires. The Purge mechanism allows for the early removal of Encoding Bundles to save storage and transmission resources in Intermediate Regulators. Purge messages MAY be ignored by Intermediate Routers, without loss of correct transmission. Purge messages may be initiated either by the Encoder or Decoder and travel along the path from the message sender to the message destination. Because the DTN path form the Encoder to Decoder may be different than the path from Decoder to Encoder, initiating Purge messages only from the Decoder may not be adequate to reach all the Intermediate Encoders

that have a copy of the Encoding Bundles for a Data Object. Purge messages MUST be authenticated before triggering the removal of Encoding Bundles, in order to avoid denial of service attacks. The Purge message MUST expire not later than the original Data Object.

Data Object Status messages that are exchanged between neighbors to indicate the availability of Encoding Bundles for Data Objects. During a contact in an opportunistic DTN network topology, neighbors must quickly determine which Bundles to exchange. Erasure Encoding helps this process, by allowing reasoning about a group of bundles based on Data Object UUID. The existence of a Data Object, rank of it's Encoding Set, and level of redundancy may be exchanged between neighbors. This information may be used to increase the efficiency of the exchange. For example, the following exchange policies MAY be enacted. If the neighbor already has the full Data Object, no bundles from that group need to be exchanged. Likewise, if the rank of the Encoding Set is almost full, then only missing Encoding Bundles need to be sent during the Innovative Send Phase. Redundant Encoding Bundles may be delayed until after other Data Objects are completed their Innovative Send Phase.

6.4. Intermediate Recoder

Early research suggests that sending Encodings that are Redundant but not Duplicates will increase the probability of receiving a full Encoding Set at the destination. In the case where the DTN topology is highly dynamic with contact transmissions much smaller than a whole Data Object, the interactions along the intermediate path mixes and copies Encodings in flight so that receiving Duplicates is common. The overall receive process can be similar to a random draw from the senders Encoding Set with replacement. Avoiding Duplicates being propagated on alternate paths avoids the "coupon collector problem" at the receiver. That is, there is a low probability of getting the last missing coupon that is not a duplicate of a previous coupon. If Duplicates are allowed to propagate on intermediate paths, many Encodings must be received before the Encoding Set can be solved, thus wasting bandwidth. The goal of the Intermediate Recoder is to introduce Redundant Encodings on the intermediate paths that are not Duplicates.

An Intermediate Recoder includes all of the functionality of an Intermediate Regulator. The Recoder adds the functionality of generating additional Encoding Bundles for Data Objects. Intermediate Recoders do not directly forward Encodings that they receive, but would produce a new Encoding from all of the previously received Encodings for the same Data Object. This new Encoding would

be Redundant relative to the previously received Encoding Set, but it would not be a Duplicate of any Encoding in the Encoding Set. Thus as the Encodings for a Data Object are propagated over the DTN, few if any Duplicates would be present over the whole DTN. In other words, if Intermediate BPAs transmitted copies of received Encoding Bundles, then Bundles that took different paths might be Duplicates. If two Duplicates are received, then at least one will be Redundant. But if two Recoded Encodings are receive, even if they come from the same partial Encoding Set, there is a chance that both will be Innovative.

Some FEC schemes allow the generation of Redundant Encodings from previously received Encodings. The exact mechanism for the generation is FEC scheme dependent, but the basic mechanism is to linearly combine multiple Encodings to form a new Recoded Encoding. The Recoded Encoding will always be Redundant relative to the Encoding Set. It adds no innovative information about the Data Object's Chunks. Instead the Recoded Encoding is not a Duplicate of previous Encodings.

More research is necessary to determine the mechanisms for Recoding. The Erasure Coding Extension supports this research by defining the architectural functions and basic policies for Recoding. Some issues to be addressed include: how to recode without raising the hamming weight; how to reduce the number of Encoding Data XORs during generation of a new Encoding; how to reduce the XORs during solving the Encoding Set; and how to generate a new Encoding that can be authenticated as if it is coming from the original Encoder.

7. Bundle Forwarding Layer

The goal of the Bundle Forwarding Layer is to transfer Bundles using all the DTN features and services available to the BPA. Legacy BPAs that are not aware of the Erasure Coding Extension MUST still forward Bundles between components, ignoring the Erasure Coding Extension, but adhering to Bundle forwarding procedures. DTN Bundle Forwarding is adequately defined and characterized in other documents and does not have too be defined here.

In order to meet the Transfer Specification at the Data Object layer, the special characteristics of the particular DTN network must be taken into account. DTN can support networks that have extreme communication models, including the following: Besides the high delay-bandwidth products found in space networks and the non-contemporaneous end-to-end paths found in opportunistic mobile networks, DTN can support transmission to selective destinations. The DTN transmission links between coding components can be either unicast or multicast. DTN supports bundles being delivered to multiple destinations. DTN also supports multiple routing protocols, that can flood bundles through out the DTN network. These DTN features may be used by Erasure Coding to support a wide variety of end-user applications. For example, reliable multicast of large Data Objects over a DTN is an illustrative use case for Erasure Coding, because FEC is an effective technique for overcoming the difficulty of feedback between the multiple receivers and the sender, and FEC may exploit the ability of the DTN to deliver Encoding Bundles among peers.

8. Security Considerations

The basic Erasure Coding protocol does not provide authentication for the Encoding Vector or the Encoding Data. This means that a rogue entity on the path between the sender and the receiver could view, delete, reorder, copy, modify, and inject new Encoding Bundles into the bundle flow. In this section we describe how to overcome this issue using the Bundle Security Protocol (BSP) [RFC6257].

The Encoding Vector is stored in the Erasure Coding Extension Block and the Encoding Data is stored in the Payload Block. The only method available in BSP to authenticate both the Payload Block and an Extension Block is through the use of a Bundle Authentication Block (BAB). This is a symmetric key-based algorithm, meaning both parties must share a secret bit-string that is not known to any other entities. BSP does not specify the method in which this secret bit-string should be established.

It should be noted that the Extension Security Block (ESB) option in BSP does not provide authentication of an Extension Block. Since ESB utilizes AES in Galois/Counter Mode (GCM), it does provide data integrity. If the AES-GCM key was the output of a key agreement protocol that authenticated the sender of the bundle, then AES-GCM encryption may provide implicit authentication. However, the ESB-RSA-AES128-EXT cipher suite that ESB utilizes does not provide this authentication.

Another issue is guaranteeing the authenticity of feedback messages (Section 6.3) generated in by the destination. This issue is not resolved because the actual need and format of the feedback messages is not defined in this document.

9. Refinement of the Erasure Coding Extension

This document defines the framework for an Erasure Coding Extension for DTN. The document defines the terms, function, architectural components, and the extension block format. Some details of some features have be purposely left out, because we expect that the Erasure Coding Extension will have multiple options for these features. Specifically, a FEC scheme has not been defined in this document. The companion document [RandBinary] defines the specification for the Random Binary Forward Error Correcting Scheme. Other FEC schemes will be added in the future. Also, the format for Data Objects has not been defined in this document. The companion document [EcObjects] defines the specification of File and Large Bundle Data Objects. Other Data Object formats will be added in the future and each will have its own document. We also anticipate a refinement of the Handling Specification and the forwarding and traffic shaping policies of Intermediate Regulators. While this document recommends default policies, other Handling Specification definition will be added in the future and each will have their own document.

10. IANA Considerations

Erasure Coding extension defines the following well known numbers:

Bundle Protocol extension block number is the constant 0xEC.

Identifier numbers for FEC schemes in extension block are defined in [RandBinary] and future FEC scheme documents.

Identifier numbers for Data Object Format type in extension block are defined in [EcObjects] and future Data Object Format documents.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4838] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and H. Weiss, "Delay-Tolerant Networking Architecture", RFC 4838, April 2007.

[RFC5050] Scott, K. and S. Burleigh, "Bundle Protocol Specification", RFC 5050, November 2007.

[RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, August 2007.

[RFC6256] Eddy, W. and E. Davies, "Using Self-Delimiting Numeric Values in Protocols", RFC 6256, May 2011.

11.2. Informative References

[EcObjects] Zinky, J., Caro, A., and G. Stein, "Bundle Protocol Erasure Coding Basic Objects", draft-irtf-dtnrg-zinky-erasure-coding-objects-00 (work in progress), Aug 2012.

[Erasure_Wang] Wang, Y., Jain, S., Martonosi, M., and K. Fall, "Erasure-coding based routing for opportunistic networks", ACM SIGCOM Workshop on Delay-tolerant networking (WDTN 05), Aug 2005.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.

[RFC5053] Luby, M., Shokrollahi, A., Watson, M., and T. Stockhammer, "Raptor Forward Error Correction Scheme for Object Delivery", RFC 5053, October 2007.

[RFC5170] Roca, V., Neumann, C., and D. Furodet, "Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) Schemes", RFC 5170, June 2008.

[RFC5445] Watson, M., "Basic Forward Error Correction (FEC) Schemes", RFC 5445, March 2009.

[RFC5510] Lacan, J., Roca, V., Peltotalo, J., and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes", RFC 5510, April 2009.

[RFC6257] Symington, S., Farrell, S., Weiss, H., and P. Lovell, "Bundle Security Protocol Specification", RFC 6257, May 2011.

[RandBinary] Zinky, J., Caro, A., and G. Stein, "Random Binary Coding Scheme for Bundle Protocol", draft-irtf-dtnrg-zinky-random-binary-fec-scheme-00 (work in progress), Aug 2012.

Authors' Addresses

John Zinky
Raytheon BBN Technologies
10 Moulton St.
Cambridge, MA 02138
US

Email: jzinky@bbn.com

Armando Caro
Raytheon BBN Technologies
10 Moulton St.
Cambridge, MA 02138
US

Email: acaro@bbn.com

Gregory Stein
Laboratory for Telecommunications Sciences
8080 Greenmead Drive
College Park, MD 20740
US

Email: gstein@ece.umd.edu