A look at the ROUTE forward

ROUTE ~ Real-time Object-delivery over Unidirectional Transport

Michael Luby

Qualcomm Technologies, Inc.

Mile High Video 2018

August 1, 2018

ROUTE

(Real-time Object-delivery over Unidirectional Transport)

- ROUTE Source Protocol
  - Broadcast/multicast source data for objects
- ROUTE Repair Protocol
  - Broadcast/multicast repair data for objects generated using RaptorQ erasure code
- HTTP Repair Protocol
  - Use HTTP to request and receive missing source data from objects

ROUTE

- **Based on FLUTE**
  - FLUTE designed for broadcast/multicast delivery of large non-real time (NRT) objects
  - Does not efficiently handle a real-time stream of smaller objects
- **Based on DASH**
  - DASH designed for HTTP delivery of a real-time stream of smaller objects
  - Not designed for broadcast/multicast delivery

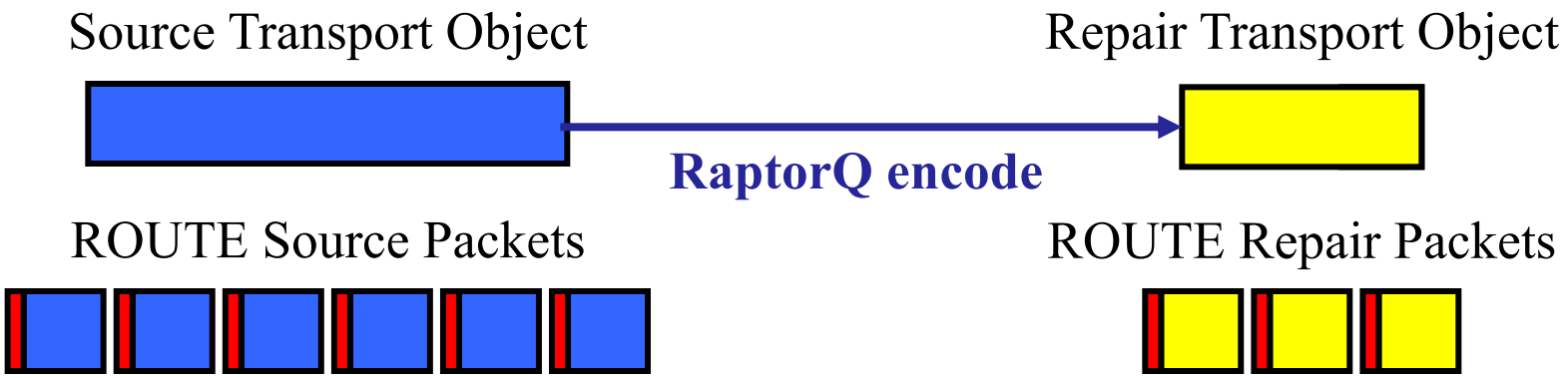
ROUTE

- **ROUTE takes from FLUTE**
  - Broadcast/multicast delivery of objects framework
  - Reliability provided by using RaptorQ erasure codes
- **ROUTE takes from DASH**
  - Rule-based signalling of parameters common to objects of a stream, e.g., template rules for URIs, timing, etc.

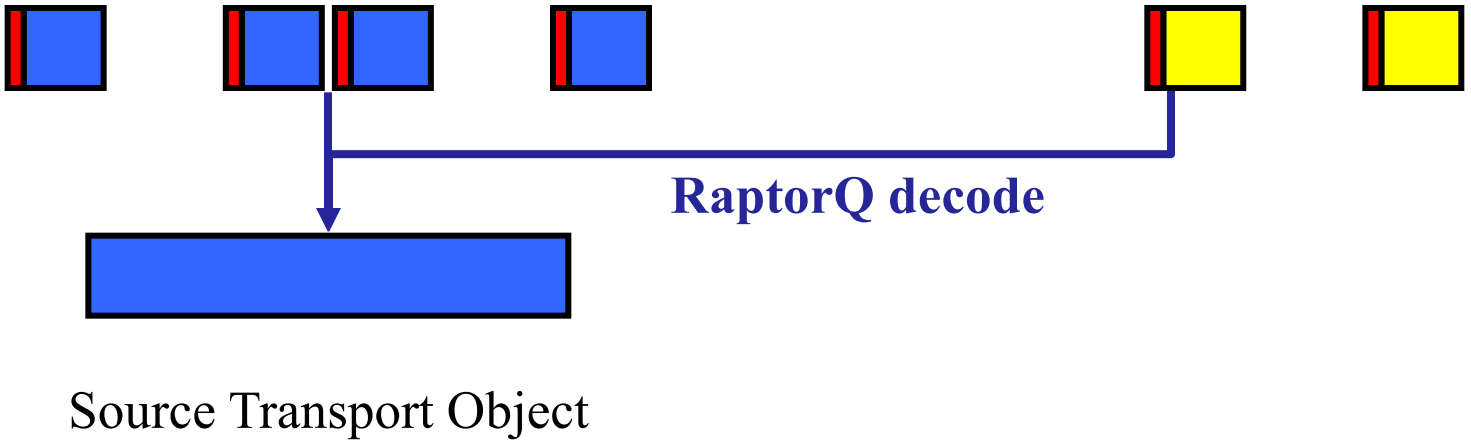
Objects

- **Application Object**
  - Has meaning to an application, e.g., a NRT file or a DASH segment
- **Source Transport Object**
  - Unit of data that ROUTE tries to reliably deliver to receivers
  - Often essentially an application object, may carry application parameters as well, can be a partial application object, or multiple application objects
- **Repair Transport Object**
  - Generated using RaptorQ encoder from source transport object(s)
  - Sent to help receivers reliably recover source transport objects

RaptorQ encoding and decoding



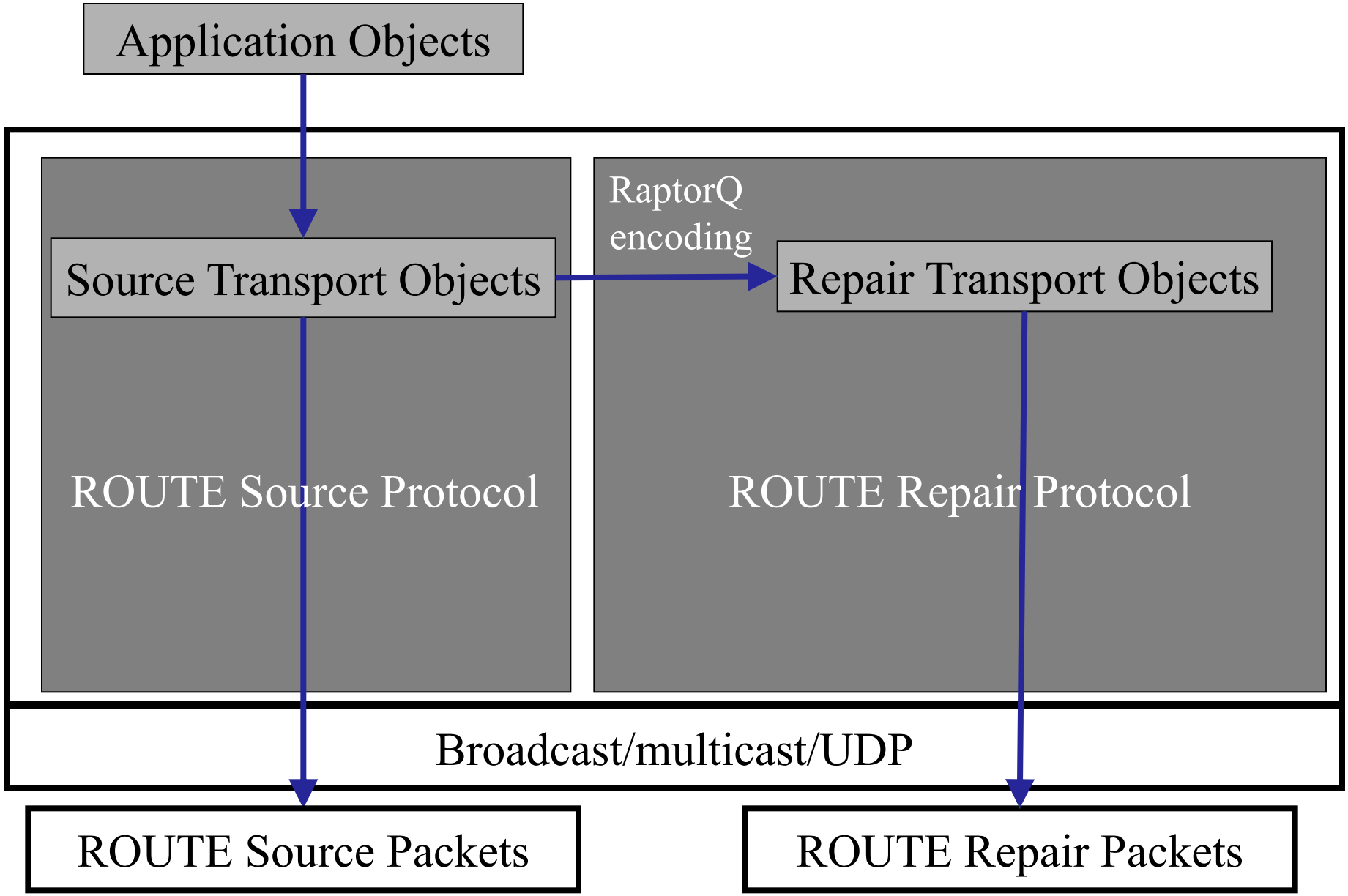
Transmit



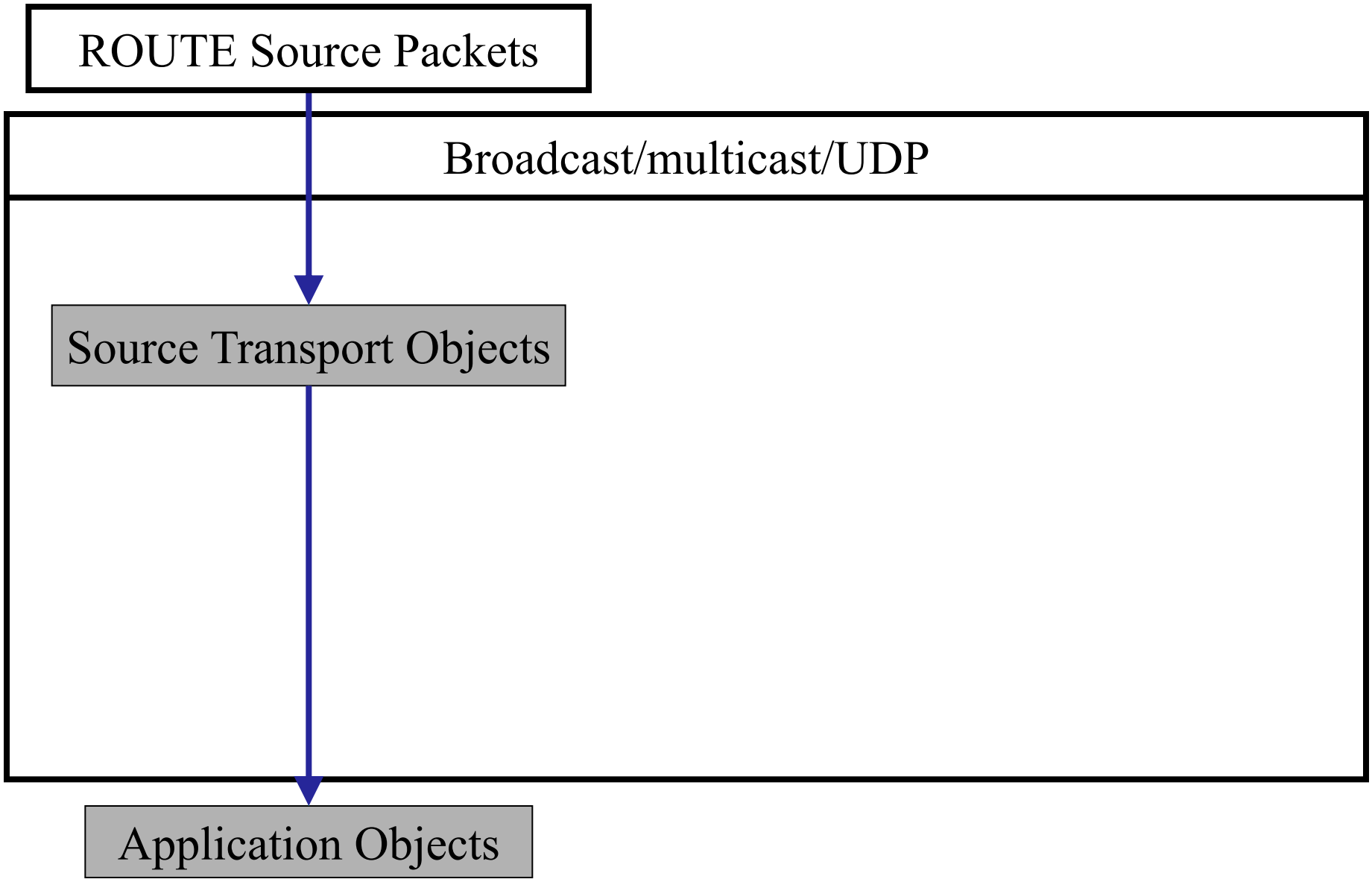
RaptorQ codes specified in IETF RFC 6330

- Low impact encoding and decoding
  - Gigabits per second encoding and decoding on standard platforms for relevant size Source Transport Object
- Essentially optimal recovery properties
- Great flexibility
  - Source Transport Object size can be tiny to large
  - Repair Transport Object size is very flexible

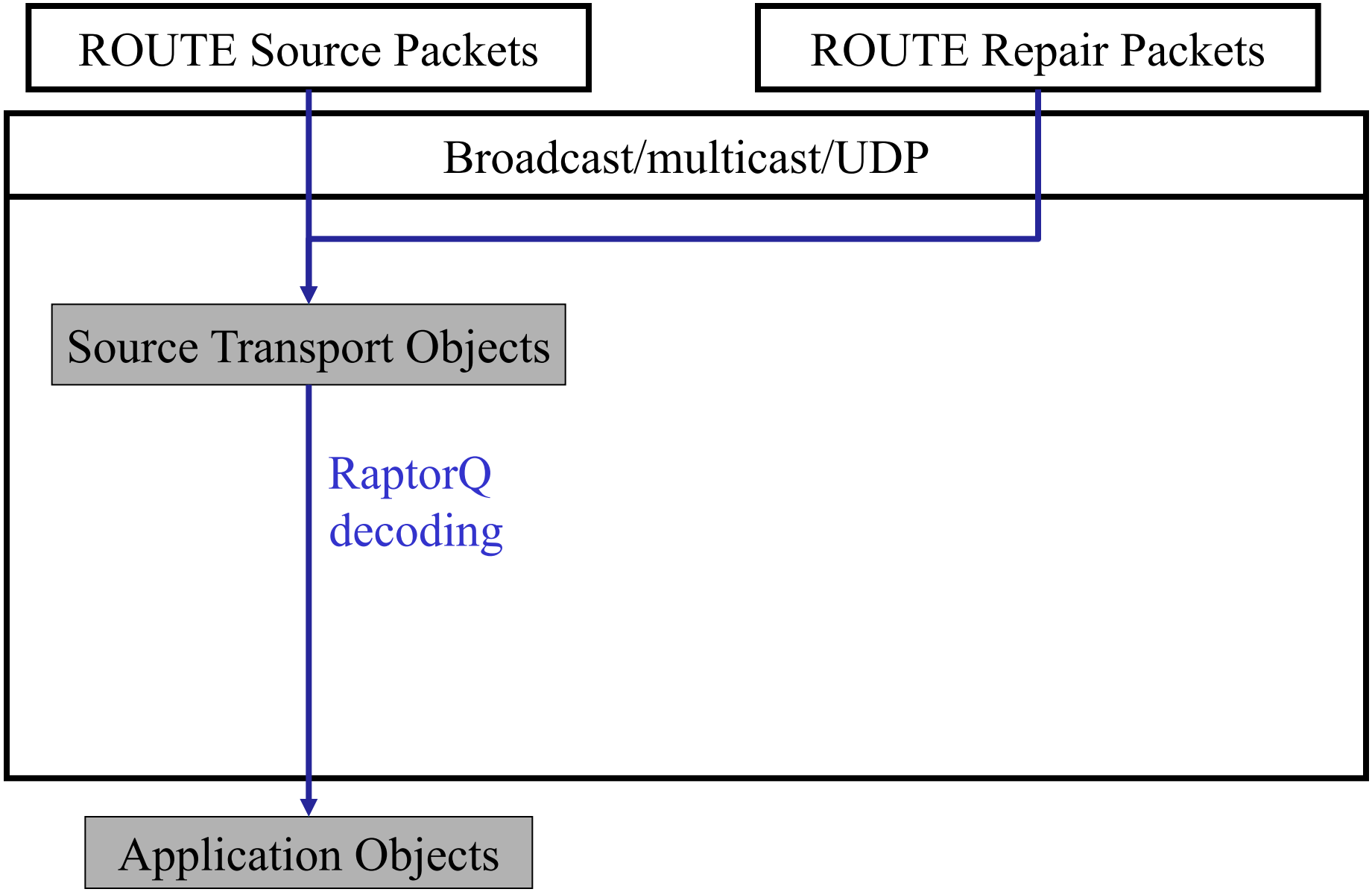
ROUTE sender



ROUTE Source-only receiver



ROUTE Fully-enabled receiver



Protocol concepts

- **Ensemble**
  - Series or set of related Transport Objects
  - Transport Object Identifiers (TOIs) are typically 1, 2, 3, 4, …
- **Transport Object**
  - Sent within an Ensemble
- **ROUTE Packet**
  - Carries data for a Transport Object

Ensemble Parameters

- **Common to all Transport Objects of an Ensemble**
  - Often TOI-based template rule
  - Fixed-size description per ensemble
  - Delivered out-of-band to receivers
    - Similar to MPD of DASH
- **Examples for a Source Ensemble**
  - Media-Type = video
    - Source Transport Objects are video segments
  - URI = “myVideo$TOI%05d$.mps”
    - For TOI=33, URI resolves to “myVideo00033.mps”
  - MaxExpiresDelta = 2.0 seconds
    - After receiving first packet for TOI=33, no more packets will be received with TOI=33 after 2.0 seconds

Transport Object Parameters

- **Vary from Transport Object to Transport Object**
- **Recovery Transport Object parameters**
  - Used to directly recover Source Transport Object
  - Delivered in-band in ROUTE packets
  - Examples include TOI, Transmission Object Size
- **Delivery Transport Object parameters**
  - Related to App Objects carried by Source Transport Object
  - Used to form, identify, provide info about App Objects
  - Delivered in-band
    - As trailer of Source Transport Object in ROUTE 2.0
  - Examples include exact size of App Object data carried in Source Transport Object, start-byte offset into App Object carried by Source Transport Object, Playback duration, URI

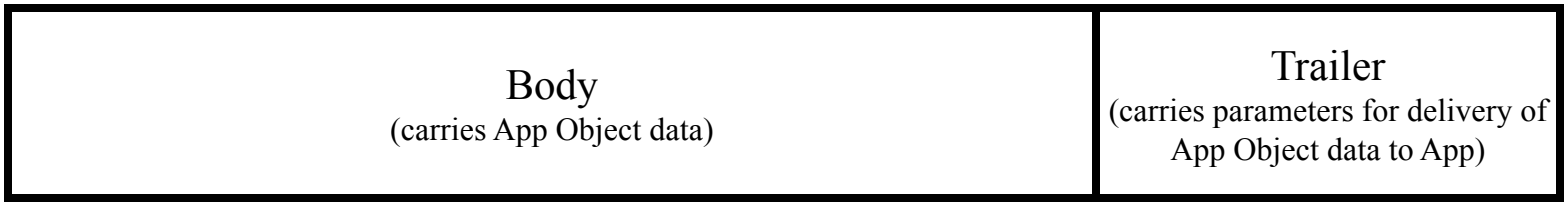
History

- **ATSC developed A-331 (ATSC 3.0)**
- **IETF ROUTE draft (in preparation)**
  - Describes overall principles of ROUTE
  - ROUTE 1.0
    - Based on and compatible with A-331
  - ROUTE 2.0
    - Based on ROUTE 1.0
    - Further simplified and cleaned up, lighter weight and expanded features
    - May be the basis for further standardization work, e.g., in DVB

ROUTE 2.0

- **Borrows concepts from HTTP Entity Mode**
  - HTTP Entity Body – data
  - HTTP Entity Header – parameters describing data
- **Source Transport Object**
  - Source Transport Body – App Object data
  - Source Transport Trailer – Delivery Transport Object Parameters

ROUTE 2.0 Source Transport Object



- Trailer – Last two fields are *F*, *A*
  - *A* is size of trailer in bytes (3 byte field)
  - *F* is size of body in bytes (5 byte field)

ROUTE 2.0 Source Transport Object Packet Header

12 bytes total

- **Payload ID**
  - Start-byte offset into data of Source Transport Object carried in the Packet Payload
- **Transport Object Size**
  - Compact representation of size of Source Transport Object
  - Includes Body, Trailer, possible padding bytes

ROUTE 2.0 Repair Transport Object Packet Header

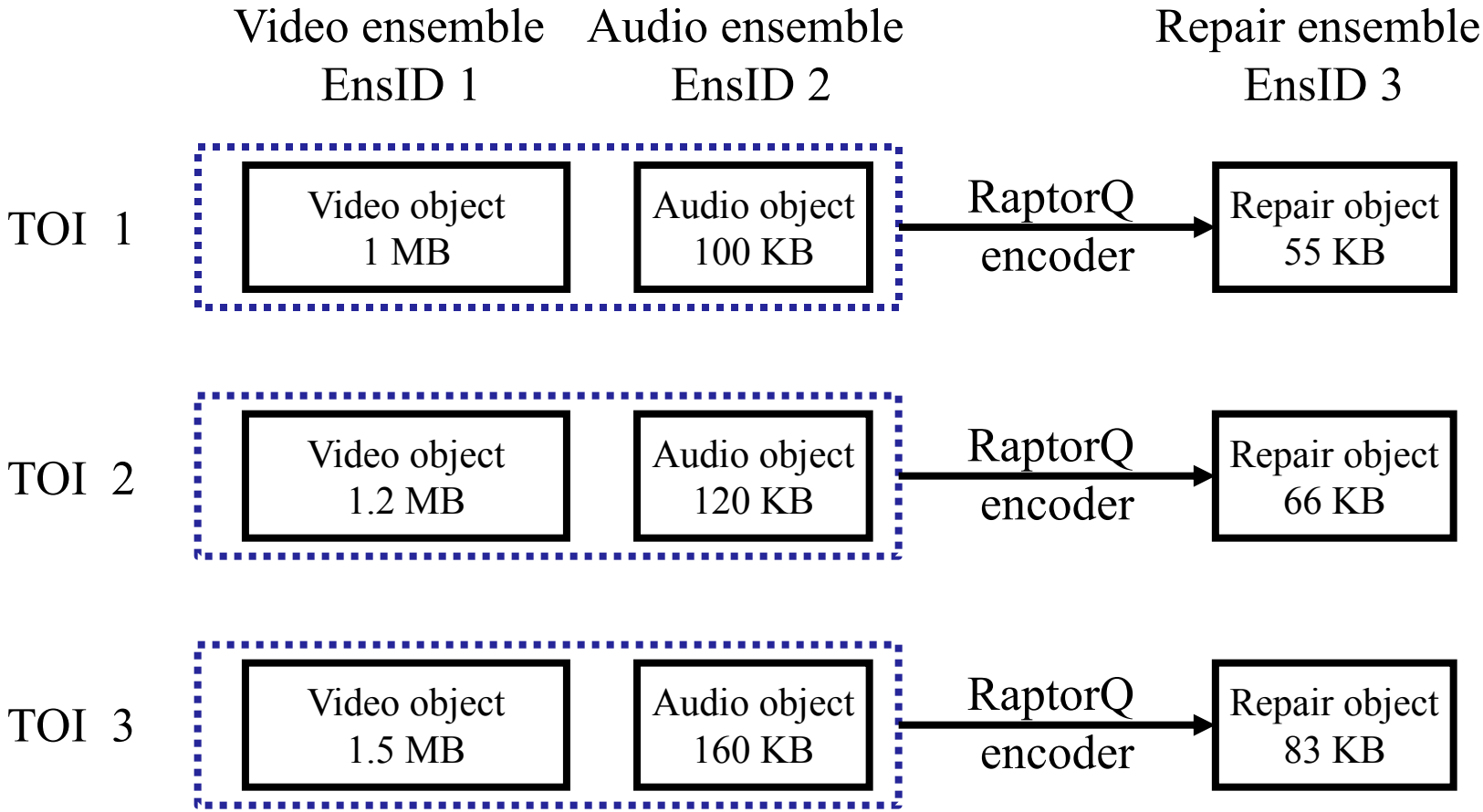
12 bytes total – when protecting one Source Transport Object

- **Payload ID**
  - Start-byte offset into data of Repair Transport Object carried in the Packet Payload
- **Transport Object Size**
  - Compact representation of size of Source Transport Object protected by Repair Transport Object

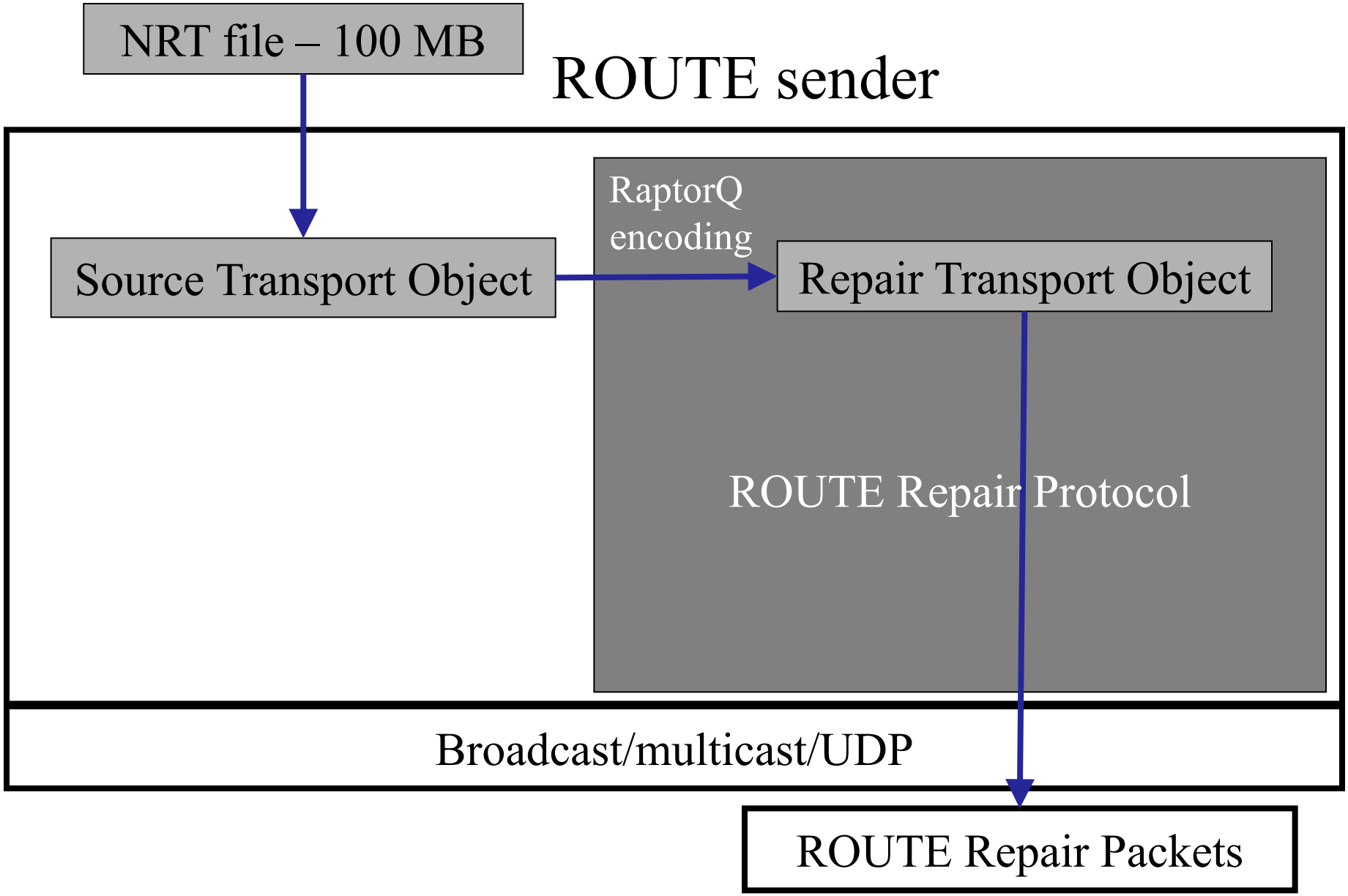
ROUTE 2.0 – streaming example

- **Source ensemble of video (DASH representation)**
  - Ensemble ID = 1
  - Ensemble parameters
    - Ensemble_Type = source, Media-Type = video, URI = “myVideo$TOI%05d$.mps”
- **Source ensemble of audio (DASH representation)**
  - Ensemble ID = 2
  - Ensemble parameters
    - Ensemble_Type = source, Media-Type = audio, URI = “myAudio$TOI%05d$.mps”
- **Repair ensemble to protect video and audio**
  - Ensemble ID = 3
  - Ensemble parameters
    - Ensemble_Type = repair, Number_protected = 2, Ens_IDs = {1, 2}

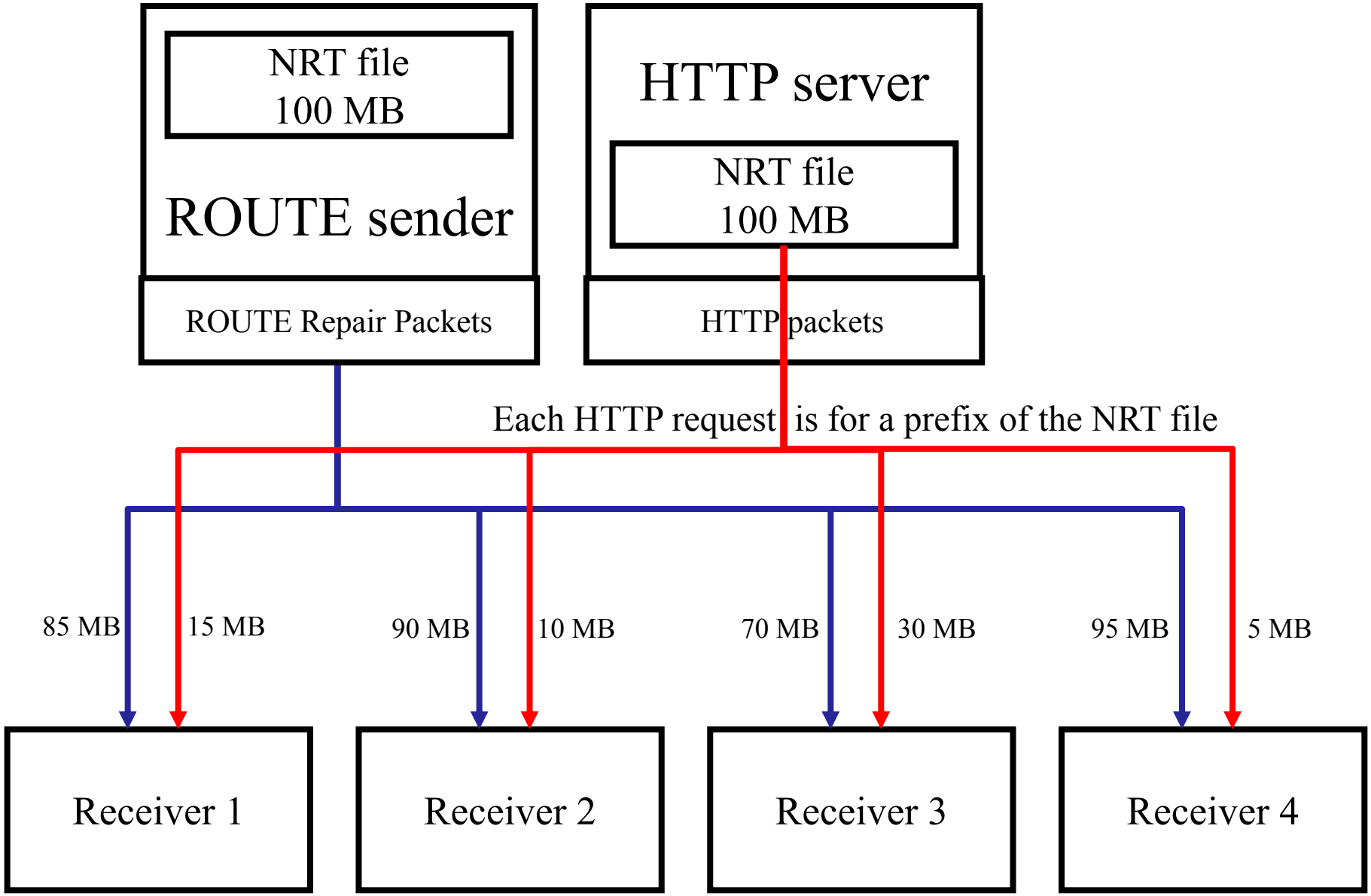
ROUTE 2.0 – streaming example



ROUTE 2.0 – NRT file example



ROUTE 2.0 – NRT file example



A look at my ROUTE forward

mgluby@gmail.com

Thanks!