Standardizing Reliable Multicast Transport Protocols in the IETF:

One-to-many Bulk Data Transfer

A Bit of History…

- Individual activities, experimental protocols and software
  - Since early 1990s
  - Various presentations, software vendors, some publications as RFCs
- Repeated calls for reliable multicast standardization
  - In the face of congestion control: not sufficiently well understood
- Reliable Multicast Research Group (RMRG)
  - http://www.east.isi.edu/rm
  - Formed in 1997, 7 meetings up to end of 1999
  - Core task: understanding congestion control for reliable multicast
- RMT WG formed in the IETF
  - Work on a well-defined (and reasonably well understood) subset
  - Basis: RFC 2357: Criteria for Evaluating Reliable Multicast Protocols
  - Simple application first: one-to-many bulk data transfer
RFC 2357 Criteria

- Scalability
- Error Recovery
- Robustness
- Security
- Congestion control, congestion control, congestion control, ...
- Simulations, protocol analysis, applicability statement

RM Design Space for Bulk Data Transfer

- TCP perceived as the general reliable point-to-point protocol satisfying many applications’ needs
  - Comparably common requirements for reliable unicast transport?!
  - Well, at least many applications can live with using TCP!
  - Development of DCCP, SCTP, and others show that needs are broader

- A single Reliable Multicast Transport?
  - Many more solutions conceivable than for unicast
  - But considered even more application-specific
    - Application requirements may impose design constraints
    - Likely to lead to only partially or non-intersecting solution spaces
  - One-size-fits-all does not exist!
“Bulk Data Transfer”

Characteristics
- Long-lived session (at least 10s of seconds)
- Continuous transmission of data
  - i.e. steady information flow rather than bursts followed by silence

Examples
- File transfer
- Continuous new feed
- Media stream

Application Constraints
- Receive confirmations?
- Limit differences between receivers?
- Scale to large numbers of receivers?
- Totally reliable?
- Ordered data? (type of ordering)
- Low-delay delivery?
- Time-bounded delivery?
- Multiple (interacting) senders?
- Intermittent flows?
- Workable on the public Internet?
- Workable without return path?
- Secure delivery?
Application Constraints

- Receive confirmations?
- Limit differences between receivers?
- Scale to large numbers of receivers? Yes
- Totally reliable?
- Ordered data? (type of ordering)
- Low-delay delivery?
- Time-bounded delivery?
- Multiple (interacting) senders? No – not well understood
- Intermittent flows? No – by definition of bulk
- Workable on the public Internet? Yes – A MUST for the IETF
- Workable without return path?
- Secure delivery?

Reliability and Scaling

- Application level (ADU) vs. packet level ("PDU")
  - Negative or positive ACKs at application potentially much less frequent
    - E.g. one per 1 GB file transfer
  - ADU may operate on different time scales
    - Feedback not needed in the order of RTT

- Scaling mechanisms
  - Positive acknowledgements (ACKs)
  - Negative acknowledgements (NAKs)
  - ACK / NAK Aggregation
  - Topologies
    - Flat often does not scale
    - Tree, ring, ...
  - Best scaling property: no feedback at all
Network Assistance

- Traditional: Keep application state out of the network
- Occasional ideas: A little bit of support would be nice
- Extreme approach: active networking (load code into routers)

Four approaches
- No support
- Layered Coding
- Server-based (server ≠ router!)
  - Servers may be senders, receivers, or some other type of entity
- Router assist
  - Attention: router state + fast path processing

RMT: Approach and Terminology

Modular design using **Building Blocks**

- Protocol family
  - Class of reliable multicast protocols with common characteristics
  - Mechanism to achieve reliability
  - NACK only, Tree-based ACK, asynchronous layered coding (ALC), Router assist
- Protocol component
  - Logical part of a protocol providing certain functionality
- Building block (BB)
  - Provides one, more than one, or part of a component
- Protocol core
  - Set of functionality required by an app but not specified by a building block
- Protocol instantiation (PI)
  - A reliable multicast protocol defined in terms of BBs and a protocol core
Protocol Components

- Data reliability (ensuring good throughput)
  - Loss detection / notification
  - Loss recovery
  - Loss protection
- Congestion control
  - Congestion feedback
  - Rate regulation
  - Receiver controls
- Security
- Group membership
  - Membership notification
  - Membership management
  - (Session management)

Building Blocks

- “a logical component that results in explicit APIs for use by other building blocks or by the protocol client”
- Suggested building blocks (for “now”)
  - NACK-based reliability
  - FEC coding
  - Congestion control
  - Generic router support
  - Tree configuration
  - Data security
- Several protocol instantiations for each BB expected
- Supposed to use/re-use common protocol headers
One-to-many Bulk Data Transfer

- Target parameters for one development
  - Receiver groups: 1,000 – 1,000,000+
  - Object sizes up to many GBs
  - One sender (by definition)
    - But receivers may obtain data from multiple senders in multiple groups
  - No need for a return channel

- Building Blocks
  - Forward error correction
  - Layered coding (LCT)
- Protocol instantiation: Asynchronous Layered Coding (ALC)
  - Uses LCT and FEC
- Specific instantiation for file transfer: FLUTE
  - Using ALC

- Note: All RFCs are still experimental!

Forward Error Correction (FEC)

- Alternative to NAK (ARQ)-based reliability schemes
- Two types of errors
  - Bit errors (may be repaired by link layer FEC)
  - Erasures (missing packets; relevant for transport layer)
    - UDP checksum turns bit errors into erasures
- Also referred to as erasure codes
- Pro-active vs. re-active
- Operating principle
Types of FEC Codes

- **Block FEC codes**
  - Operate blockwise on a number of k source symbols
  - Generate n encoding symbols for each block
  - **Systematic code**
    - leave k source symbols intact
    - generate n-k redundant symbols
  - Other codes may generate n new encoding symbols

- **Expandable FEC codes**
  - Operates on k input source symbols
  - Generates an arbitrary number of unique encoding symbols (on demand)
    - Parameterized by identifier for each unique encoding symbol

- **Simple FEC Codes**
  - XOR / Parity
    - Compute XOR over a sequence of equally long packets
    - Add the result as redundant information packet
    - 1-out-of-n erasures can be repaired
    - Useful combination with interleaving to protect against small burst losses
  - More complex variant: two-dimensional array of packets
    - More than two dimensions difficult to handle

- **Small block codes**
  - Small k and n (e.g. k,n < 256)
  - Reed Solomon
  - Vandemonde matrices

- **Large block codes**
  - E.g. Tornado codes (patented)

- **Expandable FEC codes**
FEC Building Block

- Defines how to carry FEC encoded data in an RM session
- Depends on other protocol building blocks
  - Provides only minimal header information by itself

Parameterization of FEC
- Carried in packets or out-of-band
- Registered with IANA
- FEC Encoding: which FEC encoder to be used
  - Indicates "Fully-specified" or "under-specified" encoder
- FEC Instance ID
  - Provides additional information for under-specified encoder
- FEC Payload ID
  - Identifies content of packet; specific to encoder (e.g. source info, encoding info)
- FEC Object Transmission Information
  - Additional information required by the FEC decoder (e.g. packet length, if it may differ)

FEC Header Fields

```
0                   1                   2                   3
+-------------------+-------------------+-------------------+-------------------+
| Source Block Number|                   |                   |                   |
+-------------------+-------------------+-------------------+-------------------+
| Encoding Symbol ID|                   |                   |                   |
+-------------------+-------------------+-------------------+-------------------+
```

```
0                   1                   2                   3
+-------------------+-------------------+-------------------+-------------------+
| Source Block Number|                   |                   |                   |
+-------------------+-------------------+-------------------+-------------------+
| Source Block Length| Encoding Symbol ID|                   |                   |
+-------------------+-------------------+-------------------+-------------------+
```
Layered Coding Transport (LCT)

- Target applications: reliable content delivery and streaming
- Works with SSM and ASM

**LCT session**
- A group of LCT channels associated with a single sender
- An LCT channel is identified by the sender and a multicast address
- Identifies distinct objects carried in the session

- Provides for some general header fields
- To be combined with other building blocks
  - Particularly with FEC

**Principle for Layered Coding**

- Transmission of a “content object” over more than one channel
  - Each channel identified by a (multicast) transport address
  - Receivers listen to one or more of those addresses
  - Responsibility for reliability and/or quality at the receiving side
- Prepare the object for layered transmission
  - Divide the content object into complementary parts or
  - Create multiple encodings from the content object
- Transmit distinct parts / encodings over the various channels
- Example: HDTV video, 3 layers
  - Lowest layer: b&w, mono
  - Middle layer: color, PAL quality, stereo
  - Highest layer: color, widescreen, 5+1 surround
- Similarly applicable to reliable data transmission
Protocol algorithm

- Prerequisite: session information available on both sides
  - E.g. SDP (RFC 2327), XML (RFC 3023), …

- Senders just send “objects”
  - Each object uniquely identified by Transport Object ID (TOI)
  - Transmission may be sequentially or concurrently
  - Session usually lasts for longer than a single transmission period
  - End object transmission / end of session should be announced

- Receivers listen to incoming packets
  - Validate authentication information if present
  - Demultiplex based upon header information
  - Reconstruct received objects (as soon as possible)
  - May interact with senders out-of-band

- Congestion control!

General Packet Header

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   V   | C | r |S| O |H|T|R|A|B| HDR_LEN | Codepoint (CP) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Congestion Control Information (CCI, length = 32*(C+1) bits) |
    ...                                                      
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Transport Session Identifier (TSI, length = 32*S+16*H bits) |
    ...                                                      
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Transport Object Identifier (TOI, length = 32*O+16*H bits) |
    ...                                                      
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Sender Current Time (SCT, if T = 1) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Expected Residual Time (ERT, if R = 1) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Header Extensions (if applicable) |
    ...                                                      
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
Header Extensions

Asynchronous Layered Coding (ALC)

- Defines combined use of FEC and LCT
  - The RFC numbering is quite misleading here!

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  HET (<=127)  |       HEL     |                               |
|                  |                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.              Header Extension Content (HEC)                   .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  HET (>=128)  |       Header Extension Content (HEC)          |
|                  |                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
ALC Session Description

- Congestion control BB to be used
- Sender IP address
- Number of channels in the session
- Address + port number for each channel
- Transport Session ID (TSI) for the session
- One or more objects in the session
- Format of header extensions (if any)
- Security information enable packet authentication

Summary of Reliable Multicast “Building Blocks”

- Layered Coding Transport (LCT)
  - Single sender multicast transport
  - Defines single or multi-object delivery across an LCT session
    - Provides identifiers for objects (TOI)
    - Provides session identification (TSI)
  - LCT session comprises a group of channels
    - Each identified by the respective (multicast) transport address

- Forward Error Correction (FEC)
  - General container for various FEC schemes
  - Allows to identify payload + provides in-band signaling of FEC parameters

- Asynchronous Layered Coding (ALC)
  - Simple combination of LCT and FEC
FLUTE

- File Delivery over Unidirectional Transport
- Uses ALC (= LCT + FEC)
  - Fixed parameter sets for the protocol instantiation
- Specifies semantics of objects
  - Files
  - File Delivery Table (FDT)
- FDT
  - XML-based format to carry file attributes (name, location, size, etc.)
    - Carried as Transport Object ID = 0
  - Transmitted in a carrousel style together with files

FLUTE FDT

- XML-based structured information
- Example
  
  <FDT-Payload Expires="<date>" complete="true">
    <File
      Content-Location=
      TOI=
      Content-Length=
      Transfer-Length=
      Content-Type=
      Content-Encoding=
      Content-MD5=
      ... plus some FEC stuff ... >
    <File ...>
    ...
  </FDT-Payload>