URLs and HTTP Response Forms for Multicast

`draft-singer-appsawg-mcast-url-00`

IETF 92 – Mar. 2015

David Singer and Ali C. Begen

singer@apple.com, abegen@cisco.com

Multicast Delivery Protocols: System or Transport?

- Traditionally viewed as a system (vertical)
- Yet we could view them as a transport (horizontal)
- What does it take to view them as a transport protocol that fits in?

- Note: This question is now being looked at by 3GPP SA WG4

Fitting into the Transport Landscape

- Available resources are ideally addressed by URLs
- The 'threads' of the Web *are* URL references; we can also redirect into a multicast
- Relative URLs are important for dynamism (e.g., rehosting)
- If we use DNS names we can also get some dynamic capability
- It is also really convenient to get explicit status (HTTP-like); we can then re-direct out of multicast
- Given these, it could have system software support with very similar API interface to HTTP

What Does a URL Form Need?

- Today multicasts are addressed by 'header files' (e.g., SDP)
- Classes of information in them:
  1. needed to *get going* (addresses etc.)
  2. needed to know whether it's *worth trying* (e.g., availability period)
  3. alternatives
  4. repair services, feedback addresses, etc.
- Propose we focus on the first two: the minimum 'hook'
- Other info can be in higher layer (e.g., alternatives in a manifest or program guide), or carried in-band (e.g., repair information, feedback addresses)

Requirements for the URL Form

- Data needed by ALC/LCT (see the RFCs 5775 and 5651) to *get going*
- Validity information on whether it's *worth trying* (time period, ideally network where the multicast is available)
- URLs must be valid to the specifications and support relative forms
- Steer clear of query and fragment parts (we do not own them)

Draft Form

- {prefix}{mid}{suffix}
- prefix:

```
fcast://destination:port/source:TSI
```

- mid: /name:value pairs e.g., start-time
- suffix: /label: and the URI label of the file in the multicast
- Example:

```
fcast://232.0.0.1:5620/broadcast.example.com:527353
/start:35776638264/network:media.example.com
/label:http://news.example.com/stuff.mp4
```

- We could leverage DNS for at least `destination`, `source`

Result Codes

- Fcast metainformation has HTTP headers unfortunately *EXCEPT* the result code
- We propose a new metainformation format that puts the status line (including result code) back
- Example useful codes:

  Success (the entire file is here)

  Partial content (here is a byte-range)

  Redirect (permanent or temporary)

  Not found (i.e., not here and won't be)

Daemon (Rough) Operation

- The daemon/API can ‘look like’ HTTP
  - Tune in the multicast if not currently open
  - Cache material that isn’t yet asked for
  - (Obviously) return answers as they become known (data is complete, the channel indicates a known not-found or redirect, etc.)
  - Use in-band material that is about the multicast (repair, alternatives, etc.)
  - Time-out the caching and the requests appropriately

Open Questions

- Details, escaping, exact syntax
- What is in which layer:
  - in the document enclosing the URI
  - in the URI
  - carried in-band in the multicast
- Leveraging DNS
  - Makes it more readable, more manageable, and easier to vary the answer by network (e.g., as used in some load balancers) “for you on that network, use this”
- How to constrain relative URLs (e.g., how far up can “..” go?)