RTSP

draft-ietf-mmusic-rfc2396bis-10

Magnus Westerlund

Co-auhtors:
Henning Schulzrinne, Rob Lanphier,
Anup Roa, Aravind Narasimhan

Outline

- Changes

- Open Issues

- Way Forward

- RTSP and NAT

Changes

- In summary: A LOT!

- Diff file is available at:
  http://tools.ietf.org/wg/mmusic/draft-ietf-mmusic-rfc2326bis/
  Please note that the diff file is 6 MB.

- The draft now defines RTSP version 1.1

- We have removed a lot of text having to do with backwards compatibility with pre-update RTSP agents. This text was no longer needed as we expect people to be able to implement RTSP 1.1 correctly.

- Despite that the spec has now grown to 180 pages

Changes

- Fixed some known errors that couldn't be fixed due to backwards compatibility issues:
    - RTP-Info: Now specifies SSRC and then the parameters. Thus making it possible to synchronize multiple sources.
    - Removed "destination" and "source" Transport header parameters
    - PLAY is now allowed in playing state to replace the currently ongoing PLAY action without PAUSE.
- Removed PING and put in the agreed keep-alive recommendations (SET_PARAMETER).
- Added a chapter on proxies explaining the different types of proxies:
    - Caching proxy
    - Access proxy
    - Security proxy

Changes

- Clarifications around the changing transport parameters.
- Tightened the requirement on implementation of methods. Like made PAUSE required.
- Status code 463 "Destination Prohibited" added
- Some modifications in the header tables.
- A number of syntax changes/corrections (passes ABNF parsers)
- Added clarification on remote denial of service attack in security consideration.
- Added IANA registration rules for transport header parameters.
- Added AVPF and SAVP to the RTP handling section.
- Started updating minimal implementation section.

Open Issue: State Machine

- The RTSP state machine is currently not that useful. It doesn't really provide much information on the possibilities in different states.

- One of the problems are the automatic state transfers, like when a playing RTSP server comes to the end of the media.

- Another are the synchronization ambiguities due to that client and sever are not closely coupled.

- My proposal for solution would be:
  - Specify asynchronous messages to inform the other agent about the change in state. Thus allowing for example server caused state transits while still being explicit and ensuring state consistency.
  - Specify a "state-required" header that ensure that a request is only processed if the peer state matches what is expected by the requestor. Only required to be used when necessary by request.
  - Explicit indicate the state the agent is after fulfilling a request. This ensures consistency handling of states.

Open Issue: State Machine

- Making such changes would be a major change:
    - Adds new headers carrying the state between agents.
    - Will require extra processing and some small bandwidth.
    - Implementation will more complex.
    - Requires that the state model is really explicit.

- The change would result in:
    - Would remove all ambiguities for requests that needs that explicitness.
    - Would ensure that the state is consistent after outstanding requests are resolved.
    - Would allow explicit indication of state transitions. Ensuring consistent state handling also for automatic transitions.

Open Issue:Usage of Allow header

- Allow header specifies which Methods that work with the resource
- Requested to allow inclusion of Allow in DESCRIBE and SETUP responses.
- This would remove any insecurity what methods that are allowed with resource when included.
- Possible Resolutions:
    - Make it **required** in one or the other responses as this would ensure that the client always get the information. Spends more bandwidth and processing to always determine the info
    - Make it **optional** which doesn't allow clients to rely on the possibility.
    - Make it **conditional** so it is required but only sent if doesn't match the public header.
    - **Do not allow** to not add the overhead. Only explicitly finding out of what resources are supported.

Open Issue: Minimal Implementation

- There is currently no consistency between the description of minimal implementations and the normative text.

- This section is a clear risk of inconistency within the spec. Requires a lot of work to get right.

- The section allows an implementor to be fooled by what is required. Still need to read the rest of the spec.

- The new specification is much clearer on implementation requirements.

- Proposal to remove descriptive text on what a minimal implementation requires.

- Instead only provide information on how to determine what to implement.

Way Forward

- Need to resolve the few remaining issues

- Review, more Review and some even more Review:
  - Needs reviewer that actual reads it and provide feedback
  - Need to review all aspects of the protocol, including syntax and header tables. Yes, not simple to review but it needs to be done.

- Future updates should be quite quickly forthcomming when necessary.

- Goal is to have a WG last call this year.
  - This will not happen without your help.

RTSP and NATs

- Intend to revive the RTSP NAT draft with a new version now that ICE and RTSP has made progress

- There will be need to discuss the best way of implementing ICE in RTSP.

- To start this discussion I will present a initial proposal on the next slide.

- This assumes that you have read up on ICE version 05

ICE in RTSP Proposal

Client   **Server**

Session Description and determine ICE suppport

1. SETUP with Client Candidates

2. 200 OK with Server Candidates

3. Connectivity Checks

4. SETUP to update active connection

5. 200 OK

6. PLAY

7. 200, 1xx, or 4xx