Mechanism for Diameter Overload Control (MDOC)

(draft-roach-dime-overload-ctrl)

Ben Campbell

IETF86, Orlando

Background

- Draft originally written by Adam Roach
- Adam presented it in detail in Atlanta
  - No significant change
- I'm calling it MDOC just to have something to call it :-)

- Just covering some highlights today

Piggybacked Overload Reports

- Overload Information piggybacked on other Diameter applications
  - Rate of overload reports varies with the rate of normal messaging.
  - Sends over DWR/DWA to handle quiescent connections.
- Supports overload reports in both directions for bi-directional applications.
- Reports both overload, and current load for non-overloaded nodes
- Application-ID independent.

Hop-by-Hop

- MDOC communication is hop-by-hop
  - If no agents exist, this works directly between client and server
  - If an agent exists, it consumes overload information
    - If it can locally mitigate load, it does so. If so, client never sees the overload report.
    - If it can't mitigate load locally, it originates its own overload report towards client.
      - Aggregate overload of the agent and all upstream nodes.

Hop-by-Hop (cont)

- Currently does not comply to req 35 (traversing non-supporting intermediaries)
  - We recognize this is an open issue
  - Looking into ways to make this work
- Non-adjacent reporting is complicated regardless of which mechanism is selected
  - How is it negotiated?
  - How to you avoid “over reporting” of overload?
  - May require a separate specialized mechanism.

Rich Scope Model

- Fine control over what a given overload report effects
  - Host
  - Connection
  - Destination-Realm
  - Destination-Host
  - Application-ID
  - Session
  - Session-Group

Scopes (cont)

- Scopes can be combined for fine control
  - e. g. Application-ID: X + Destination-Realm: Y + Destination-Host: X
  - Scopes are extensible. Basic scopes MTS

Session Groups

- Session-Group labeling aids in letting agents report upstream overload
  - e.g. An proxy distributes sessions among a set of servers according to some local policy.
  - The proxy adds a label to each session going to a certain server
  - If that server becomes overloaded, the proxy sends a single overload report that effects every session with that label.

Extensible Algorithms

- One MTI algorithm defined (loss)
- Overload-Metric is abstract
  - Unsigned 32 value
  - Interpretation is up to algorithm
  - For “loss” algorithm, Overload-Metric is the requested percentage load reduction