ALTO Incremental Updates using HTTP/2

Y. Richard Yang

April 21, 2020

ALTO Interim Meeting

Outline

- ALTO SSE review
- Initial design
- Discussion on next step

Review: ALTO SSE Big Picture

- Goals: (1) push updates, (2) compact/incr encoding of updates; (3) dynamic stream control
- Realization: two services
  - Update service (send update messages)
    - Data updates
    - Control updates
  - Stream control service
    - Add/remove resources receiving updates

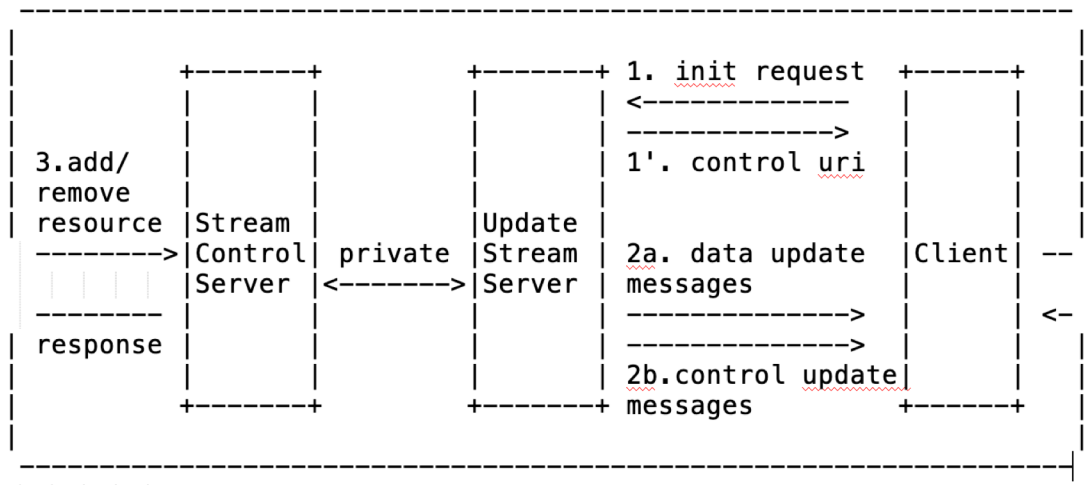
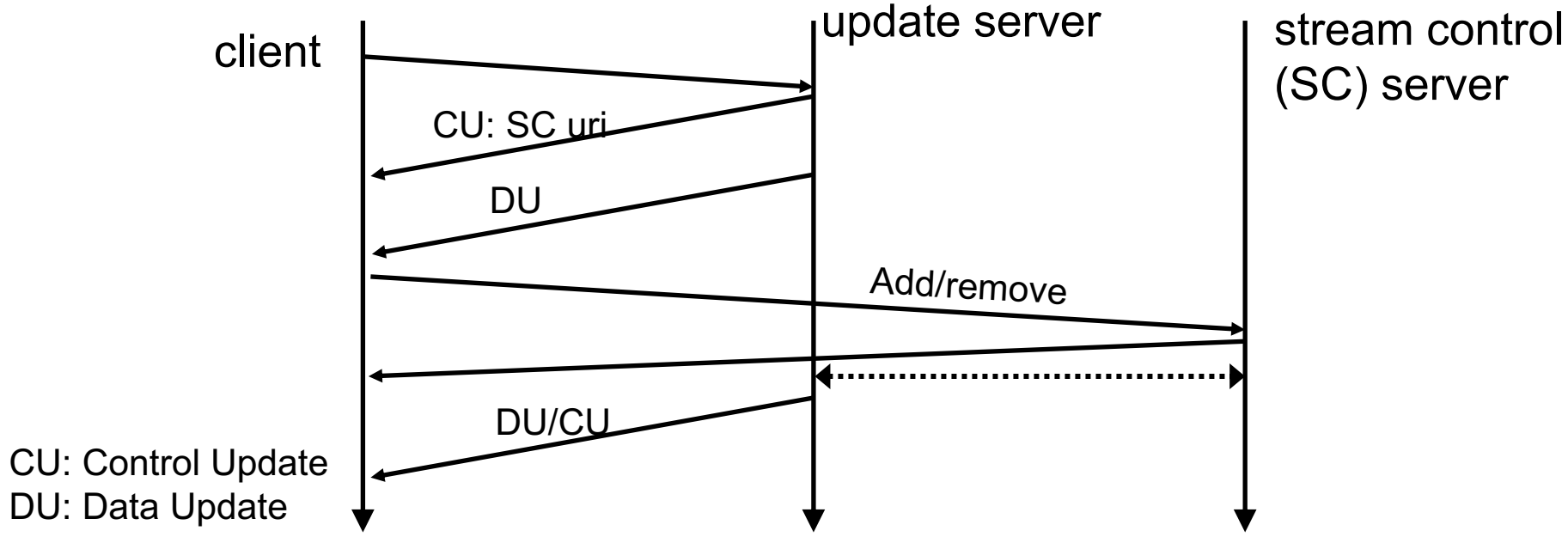


Figure 2: ALTO SSE Architecture and Message Flow.

ALTO SSE as HTTP/1.x-Compatible Design

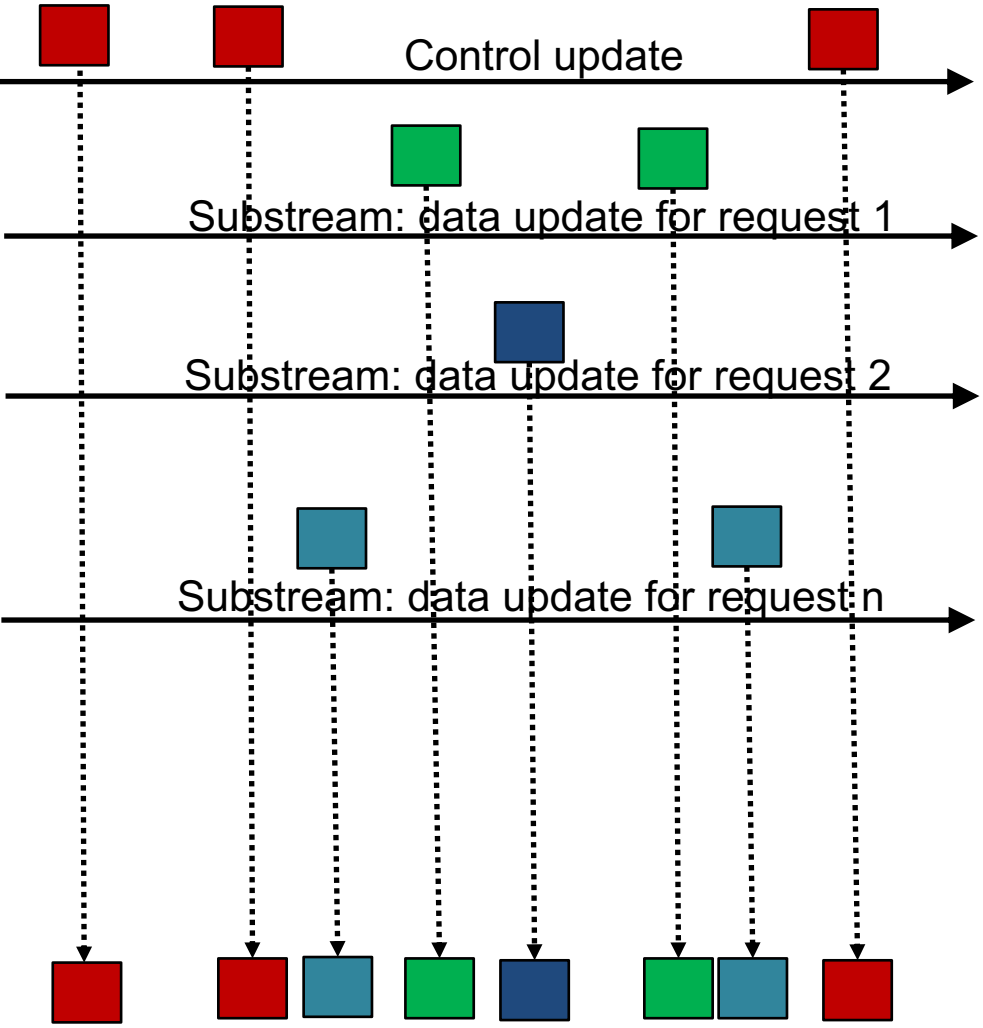
- Issue 1: Allow dynamic addition/removal of resources (called substreams) receiving updates, but HTTP/1.x allows sending only one request at a time
- Solution: two services



ALTO SSE as HTTP/1.x-Compatible Design

- Issue 2: Need to multiplex multiple logical data streams (control update, updates for different resources) with different media types (full encoding, different incr) from update server to client
- Solution: adapt existing server-sent events (SSE)
  - event: media-type [',' data-id]
    - media-type
      - control update: application/alto-updatestreamcontrol+json
        - » first update must be control update, w/ control URI
      - data update: full replacement (e.g., application/alto-networkmap+json) or incremental encoding media (e.g., application/merge-patch+json)
    - data-id (only for data update): substream-id [. content-ID for multipart/related]
      - Consider the whole connection as an update stream, and hence each data update stream and the control update stream are considered as individual sub-streams
  - data: JSON object of the given media type in the event field

HTTP/1.x Update Stream Serialization



Among update messages

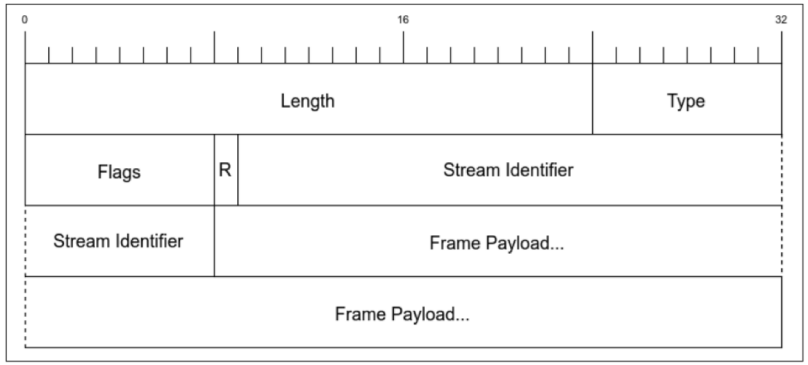
- Intra-substream
  - Update i based on Update i-1 delivered reliably, in order
- Inter-substream (e.g., CostMap depends on NetMap):
  - Conceptually can be asynchronous substreams of events, as a client can use the dependent tags to compute update ordering
  - SSE recommends that the server send being-dependent updates (e.g., NetMap) before sending dependent (e.g., CostMap) updates

Benefits of HTTP/2 Based Design

- Leverage the more modern HTTP transport
  - multiplexing
    - SSE enforces a single serialization of all substreams
      - Assume: two independent network maps have changes at the same time, SSE will still need to serialize the updates (potentially longer update latency)
    - HTTP/2 to allow concurrent updates
  - bi-directional
    - Instead of two services, we may reduce to a single service
  - More efficiency (e.g., header compression) and flexible control (e.g., priority)

Initial Design (Maximize Compatibility)

- A single HTTP/2 connection: stream control (add) and data update of each resource <-> HTTP/2 stream
  - Request: client picks HTTP/2 stream-ID (only number, no longer generic string), and sends the update request to the server
    - First control update null uri
  - Updates: server uses SSE encoding to push full-replace/incr of the resource through the HTTP/2 stream
    - event: only media-type; stream id is carried by frame; add content-id to handle multipart
    - Server handles dependency



  - Close
    - Server closes stream by indicate END_STREAM flag of last DATA
    - Client closes stream by sending RST_STREAM

Next Step

- Initial draft to be uploaded
- Feedback on the initial design highly welcome
- Focus on HTTP/2 or HTTP/2 and HTTP/3?