ALTO New Transport Extension

Kai Gao

Nov 10, 2020@ALTO Weekly Meeting
Motivation

- HTTP/2 (RFC 7540) and HTTP/3 (RFC-to-be) improve the performance of HTTP
  - Avoid slow-start of multiple connections
  - Avoid head-of-line blocking of request pliplining (in HTTP/1.1)
  - Allow concurrent transmission (HTTP/3)
  - Native support for server push and stream multiplexing

- ALTO extensions have workarounds that are unnecessary and inefficient when using HTTP/2 and above
  - Path vector
    - use multipart to encode two resources in the same response
  - Incremental update
    - allow one server to provide updates of many resources (to avoid creating too many connections)
    - use SSE to multiplex the updates
Current issues

- Transport mechanisms in the current ALTO framework
  - Path Vector
  - Incremental Updates
  - Multipart
  - SSE
  - HTTP/1.x

- Proposed alternative transport mechanism on top of HTTP/2 and above
  - ALTO
  - Path Vector
  - Incremental Updates
  - HTTP/2+
ALTO Transport over HTTP/2 and HTTP/3: Feasibility

• Workarounds can be replaced by standard features of HTTP/2 and above
  • Server push
    • Push (temporary) dependent data (e.g., path vector) as different streams
    • Push incremental updates of each resource as a single stream
  • Stream multiplexing
    • Native and more efficient support for event multiplexing
  • Rest stream
    • Subscription of incremental updates can be cancelled by the client using the rest stream frame

• Already supported by many mainstream programming languages (C/C++, Java, Python, Go, etc.) and network development frameworks (nginx, Apache, etc.)
Case study: Path vector

• One consideration of using multipart message is to avoid security risks when it takes a long time for the client to query the property map part.

• With HTTP/2+, an ALTO server can actively push the property part to the client, eliminating the need of a multipart message.
Multipart v.s. Server Push

HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: multipart/related; boundary=example-1;
type=application/alto-costmap+json

--example-1
Resource-Id: costmap
Content-Type: application/alto-costmap+json

{  
    "meta": {  
        "vtag": {  
            "resource-id": "filtered-cost-map-pv.costmap",
            "tag": "d827f484cb66ce6df6b5077cb8562b9a"
        },
        "dependent-vtags": [{  
            "resource-id": "my-default-networkmap",
            "tag": "75ed013bcb58f996e839582504f6228"
        }],
        "cost-type": {  
            "cost-mode": "array",
            "cost-metric": "ane-path"
        },
        "cost-map": {  
            "PID1": {  
                "PID2": ["ANE1"]
            }
        }
    }
}

--example-1
Resource-Id: propmap
Content-Type: application/alto-propmap+json

{  
    "meta": {  
        "dependent-vtags": [{  
            "resource-id": "filtered-cost-map-pv.costmap",
            "tag": "d827f484cb66ce6df6b5077cb8562b9a"
        }],
        "property-map": {  
            ",ane:ANE1": {  
                "max-reservable-bandwidth": 100000000
            }
        }
    }
}

// SETTINGS frame
// PUSH_PROMISE frame
:method: GET
:path: filtered-cost-map-pv.propmap
// promised_stream_id=2

// HEADER frame
:status: 200
content-type: application/alto-costmap+json
content-length: 46

// DATA frame
{  
    "meta": {  
        "vtag": {  
            "resource-id": "filtered-cost-map-pv.costmap",
            "tag": "d827f484cb66ce6df6b5077cb8562b9a"
        },
        "dependent-vtags": [{  
            "resource-id": "my-default-networkmap",
            "tag": "75ed013bcb58f996e839582504f6228"
        }],
        "cost-type": {  
            "cost-mode": "array",
            "cost-metric": "ane-path"
        },
        "cost-map": {  
            "PID1": {  
                "PID2": ["ANE1"]
            }
        }
    }
}

// HEADER frame
:status: 200
content-type: application/alto-propmap+json
expires: Thu, 19 Nov 2020 06:39:22 GMT

// DATA frame
{  
    "meta": {  
        "dependent-vtags": [{  
            "resource-id": "filtered-cost-map-pv.costmap",
            "tag": "d827f484cb66ce6df6b5077cb8562b9a"
        }],
        "property-map": {  
            ",ane:ANE1": {  
                "max-reservable-bandwidth": 100000000
            }
        }
    }
}
Specific problems to be addressed

- Define a unified transport mechanism for ALTO objects over HTTP/2 and HTTP/3
  - Specify unified naming and dependency indication of potentially dynamic resources
  - The current design is to reuse the idea of stream-id as in the incremental update extension

- Backward compatibility
  - Specify an extension to allow clients and servers to negotiate the transport mechanism
  - The current design is to add a new capability to the IRD, and clients must explicitly specify the new transport mechanism
Remaining issues

• Security and privacy concerns introduced by HTTP/2 and above need to be investigated
• Potential future capabilities
  • Transaction capability: The values/updates of dependent resources always belong to the same consistent snapshot.
Q & A

• Who will work on this extension
  • Kai
  • Other people are more than welcome to contribute!

• Milestones
  • A new IETF RFC
  • An initial version before IETF 110