ALTO Protocol

draft-ietf-alto-protocol-03

Richard Alimi (Ed.), Reinaldo Penno (Ed.), Stefano Previdi, Stanislav Shalunov, Richard Woundy, Y. Richard Yang (Ed.)

Grateful to contributions from large number of collaborators; see draft for complete list.

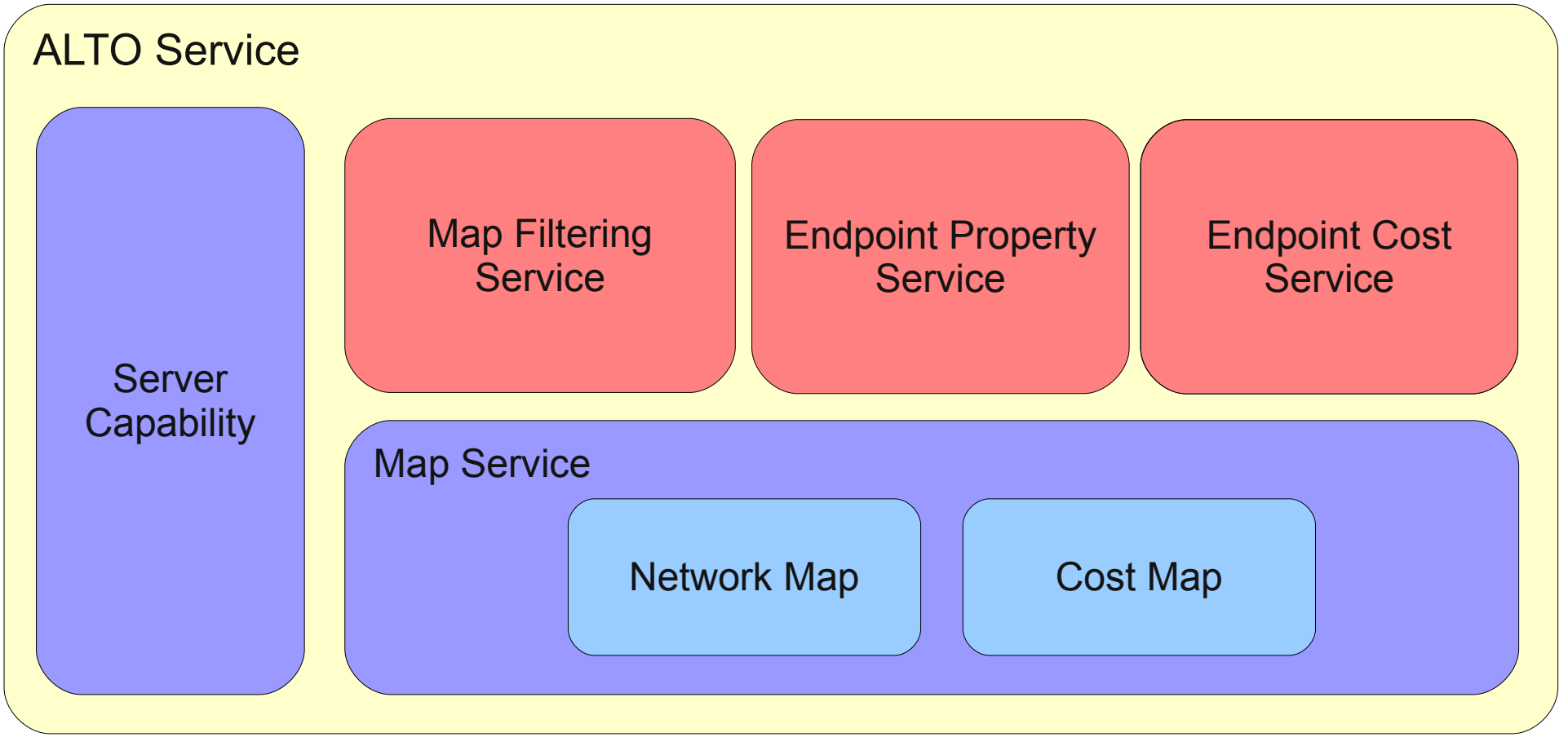
Outline

- Protocol Structure overview
- Protocol Encoding (new since IETF76)
  - Focus on major discussion points
  - Discuss specifics (e.g., particular parameters) if time permits
- Discussion

Basic Concepts (quick refresher)

- Network Locations
  - Individual Endpoints
  - PIDs for aggregation (privacy and scalability)
- Network Map
  - Mapping between Endpoints and PIDs
- Cost Map
  - Costs between Network Locations
  - Server may define multiple types of costs

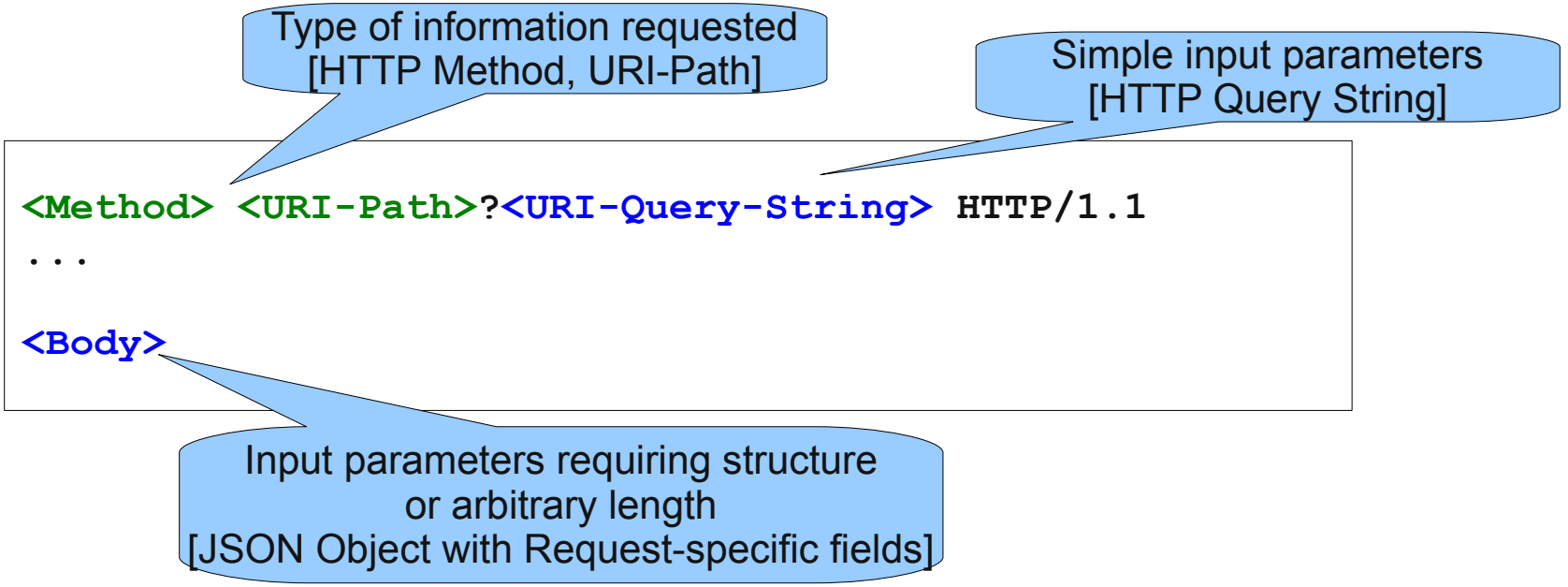
Protocol Structure



Protocol Encoding: Approach

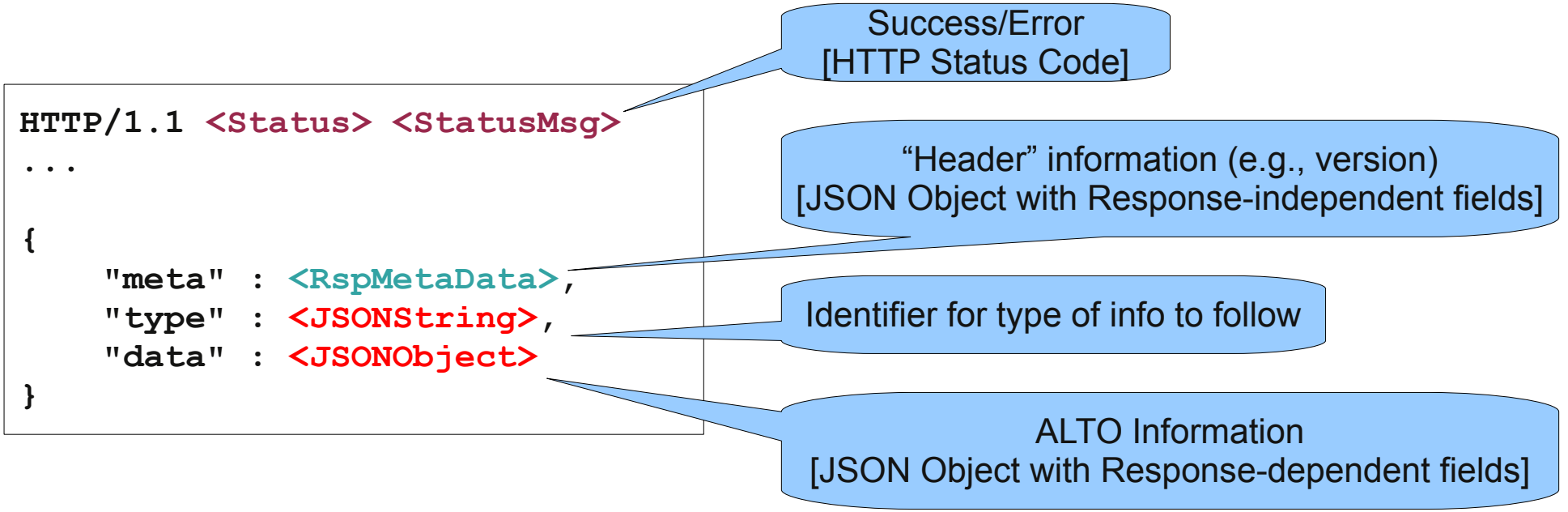
- Goals
  - Ease integration
    - Existing infrastructure (e.g., HTTP caches)
    - Many P2P apps already have an HTTP client
  - Text-encoding to ease protocol understanding/debugging
- Design Choices
  - RESTful interface over HTTP
  - JSON encoding for message bodies

ALTO Request Syntax



- Follow “standard” REST-ful design
- Approach for Input Parameters
  - Use Query String where possible and appropriate (permits caching)
  - Use Body when size of input parameters can be large or requires some structure

ALTO Response Syntax

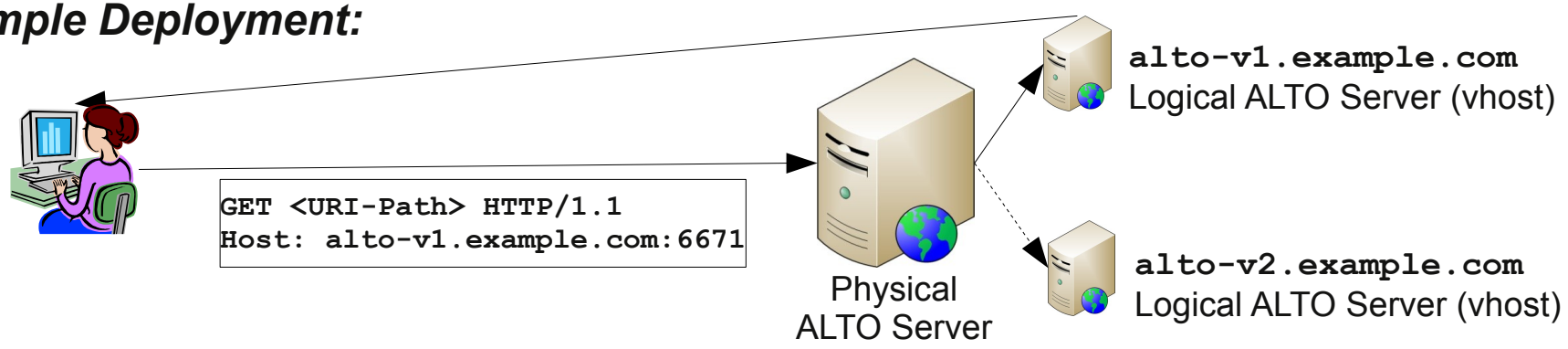


- Currently use normal HTTP Status codes
  - List discussion suggesting to (cleanly) separate application-layer status
- Body designed to be self-contained JSON Object
  - Metadata needed to interpret ALTO information stored inside body
  - Simplify persistence and redistribution

Protocol Versioning Approach

- Many REST-ful designs encode version in URI
  - Implications for Server discovery protocol, load balancing (L7 switches)
- Current approach
  - (Logical) ALTO Server implements a single protocol version
    - Demultiplexed by hostname (may use virtual hosting)
  - ALTO Client bootstraps from any ALTO Server managed by provider
    - Utilize Server Capability service (will see later...)

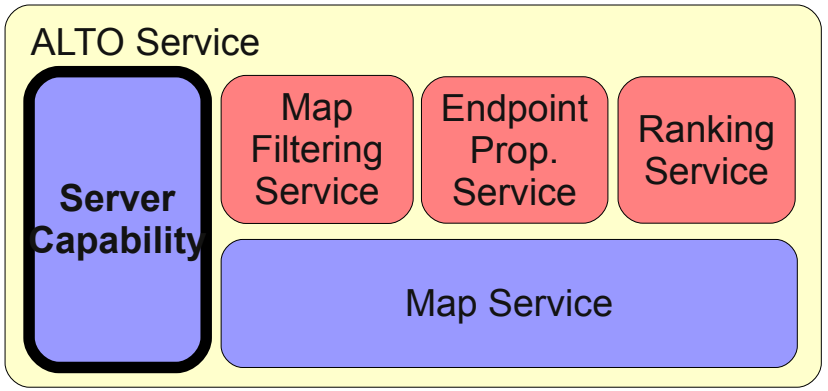
Example Deployment:



Services and Operations Overview

Service	Operation	Method and URI-Path
Server Capability	Lookup	`GET /capability`
Map	Get Network Map	`GET /map/core/pid/net`
	Get Cost Map	`GET /map/core/pid/cost`
Map Filtering	Get Network Map	`GET /map/filter/pid/net`
	Get Cost Map	`GET /map/filter/pid/cost`
Endpoint Property	Lookup	`GET /endpoint/prop/<name>`
		`POST /endpoint/prop/lookup`
Endpoint Cost	Lookup	`POST /endpoint/cost/lookup`

Server Capability



- Purpose
  - Discovery of alternate ALTO Servers (likely same administrative domain)
    - Versions, supported services, supported cost types
  - Info local to server itself
- Discussion
  - Separate query for discovery of alternate servers?
  - How much (if any) config information in discovery (“server_list”)?
  - Registry for cost types?

```
GET /capability HTTP/1.1
...


HTTP/1.1 200 OK
...

{ "meta" : ...,
  "type" : "capability",
  "data" : {
    "server_list" : [ {
        "uri": "http://alto.example.com:6671",
        "version" : 1,
        "services" : [ "map",
                       "map-filtering" ],
        "cost_types": [ ... ],
        ...
      } ],
    "self" : {
      “certificate” : “...”
    }
}
```

Changes to Remaining Services

- Renamed “Ranking Service” to “Endpoint Cost Service”
  - More accurate characterization of capabilities
- Map, Map Filtering, Endpoint Property, and Endpoint Cost Services
  - Changes since IETF76 pertain to encoding
  - Focus today's discussion on more general issues
  - … unless specific comments/feedback/questions from WG?

Redistribution

- Basic Idea (more in later presentation...)
  - Allow ALTO Clients to distribute ALTO Information to each other
    - Unit of redistribution is an ALTO Response Body
  - ALTO Clients should be able to verify authenticity of received info
- Requirements
  - ALTO Responses must identify any input parameters
    - Allows ALTO Client to identify context of received info
  - Digitally-signed ALTO Responses (by ALTO Server's private key)
    - ALTO Client can verify that response generated by particular ALTO Server
  - ALTO Client must be able to retrieve ALTO Server's public key
    - Should only need to be done infrequently

Redistribution

- ALTO Server MAY mark cachable responses as *redistributable*
  - Echo Operation and Input Parameters
    - In “redistribution” section of metadata
  - Digital signature of response body
    - In HTTP Headers/Trailers
  - ALTO Server must provide public key
    - X.509 cert in Server Capability response
- Discussion
  - Explicit distribution scope?
  - Technique other than “X-ALTO-” HTTP headers?

```
HTTP/1.1 200 OK
...
X-ALTO-HashAlgorithm: ...
X-ALTO-SignatureAlgorithm: ...
X-ALTO-SignatureDigest: ...

{ "meta" : {
    "version" : 1,
    "redistribution" : {
      "server" : "alto.example.com:6671",
      "request_uri" : "http://...",
      "request_body" : { ... },
      "expires" : "2010-03-12T23:20:50.52Z"
    }
  },
  "type" : ...,
  "data" : ...
}
```

IPv4 / IPv6

- Need to define semantics for multiple types of Endpoint IDs
  - Semantics applied to IPv4 and IPv6 is of immediate concern
  - Do we want to generalize to other endpoint identifiers?
- Discussion
  - Dual-stack hosts: can Network Provider indicate a preference? If so, at what granularity?
    - Global?
      - Example: “Always use v6 if available”
      - Cleanest approach may be separate maps each with a preference value
    - Destination Network Location?
      - Example: “Prefer v6 for Resource Providers in ISP A, but not for those in ISP B”
      - Possibility is a per-PID attribute indicating preference for v4 or v6
  - Other considerations?

Discussion

- Any other comments or feedback?