*

Distributed OAuth

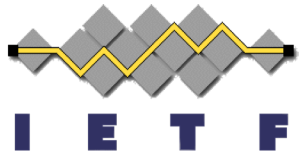
draft-hardt-oauth-distributed

Dick Hardt
IETF 102, Montreal
July 2018

Since Singapore

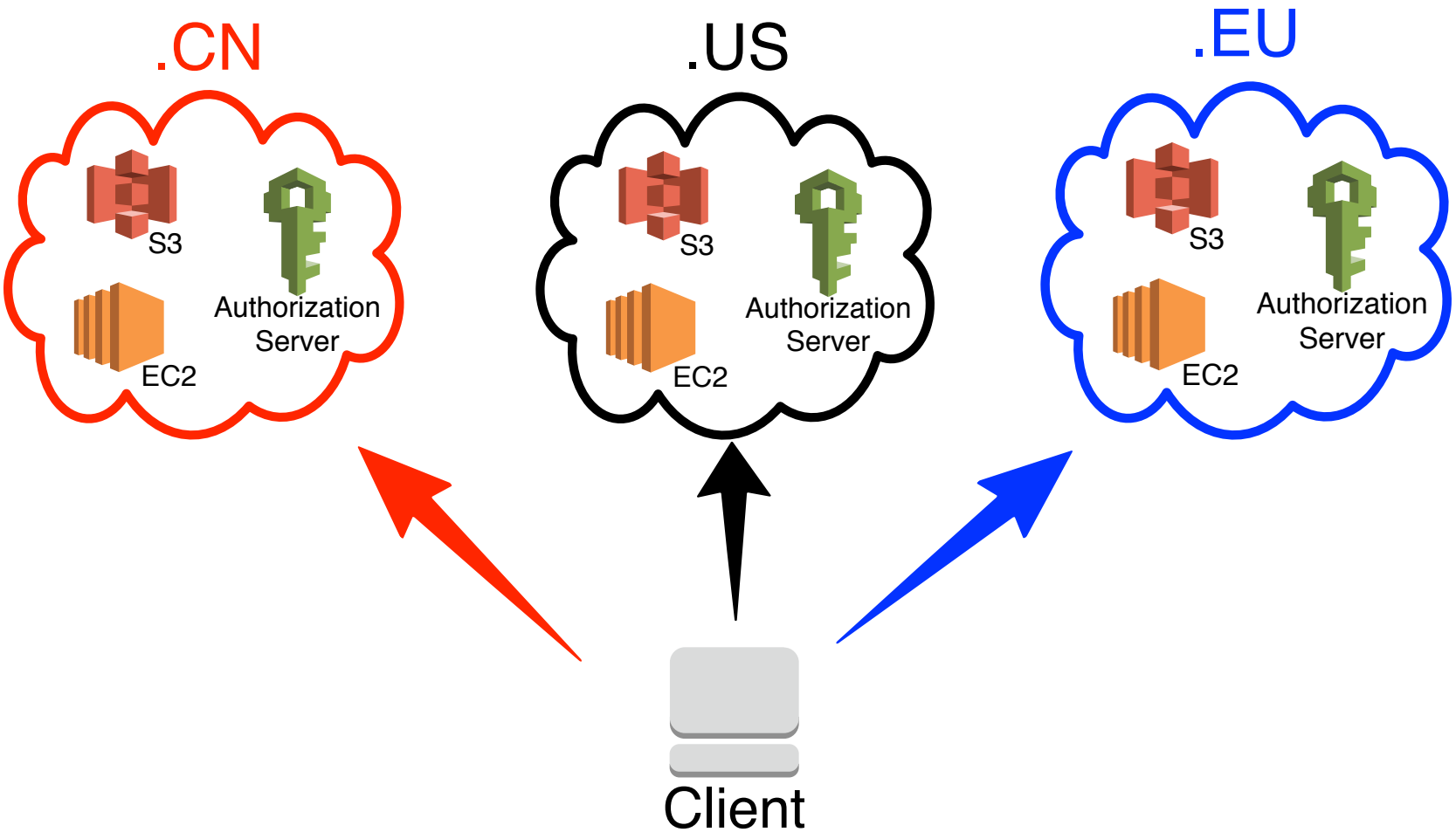
- Brian Campbell and Nat Sakimura co-authors
- Incorporated
  - draft-campbell-oauth-resource-indicators-02
  - draft-sakimura-oauth-meta-08
- -01 released
  - Resource is URI
  - All OAuth grant types supported
  - Link header used for discovery

AS Discovery Problem



- OAuth 2 presumes **static relationship** between authorization server and protected resource that is **known a priori** by client
- Global systems have similar protected resources, that are managed by different authorization servers. Eg. different geopolitical regions.
- Large, distributed systems need to evolve the relationship between authorization servers and protected resources.
- Clients need to **dynamically** learn the authorization server for a given protected resource **at run time**.

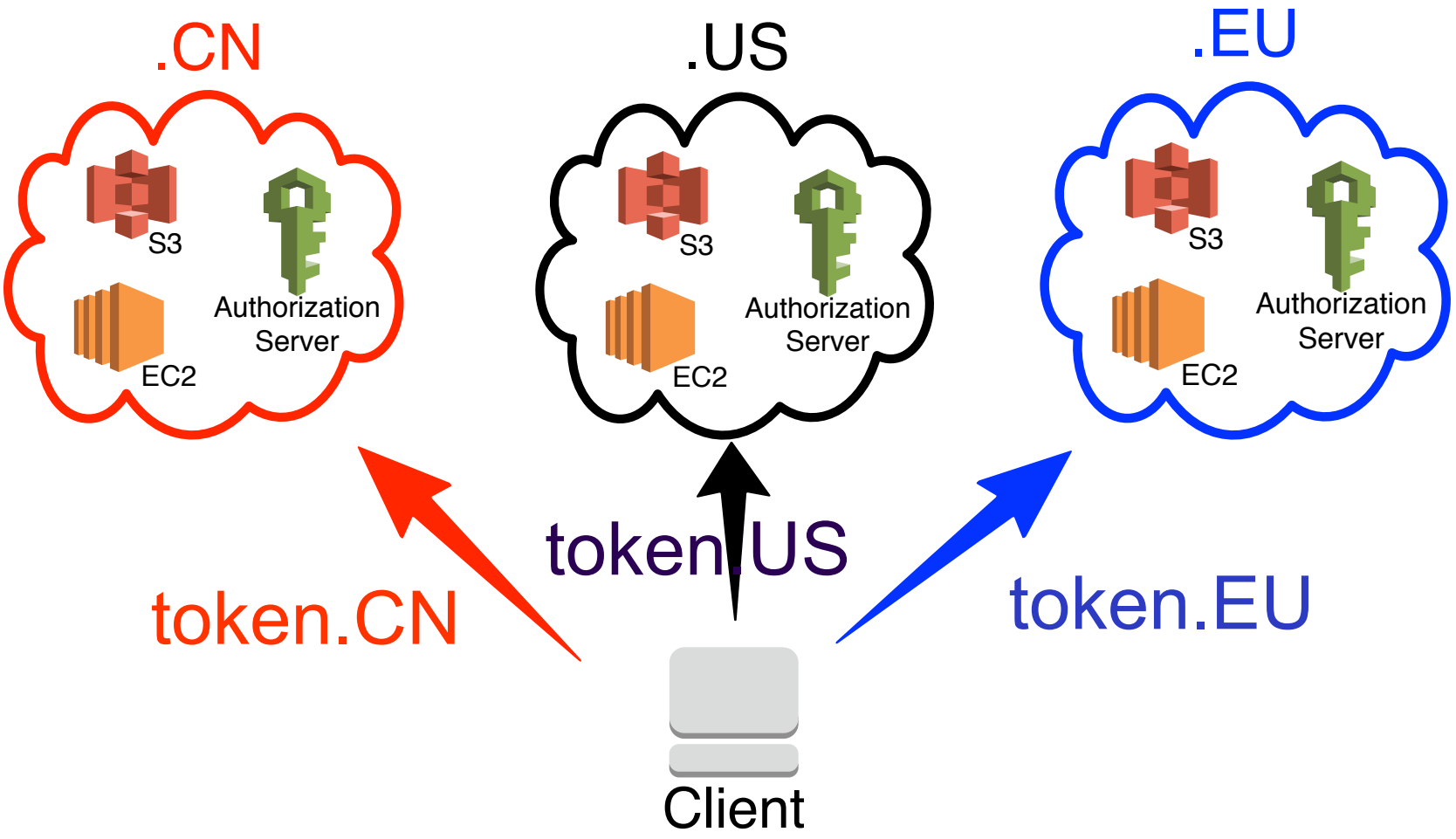
Client Accessing Global Protected Resources



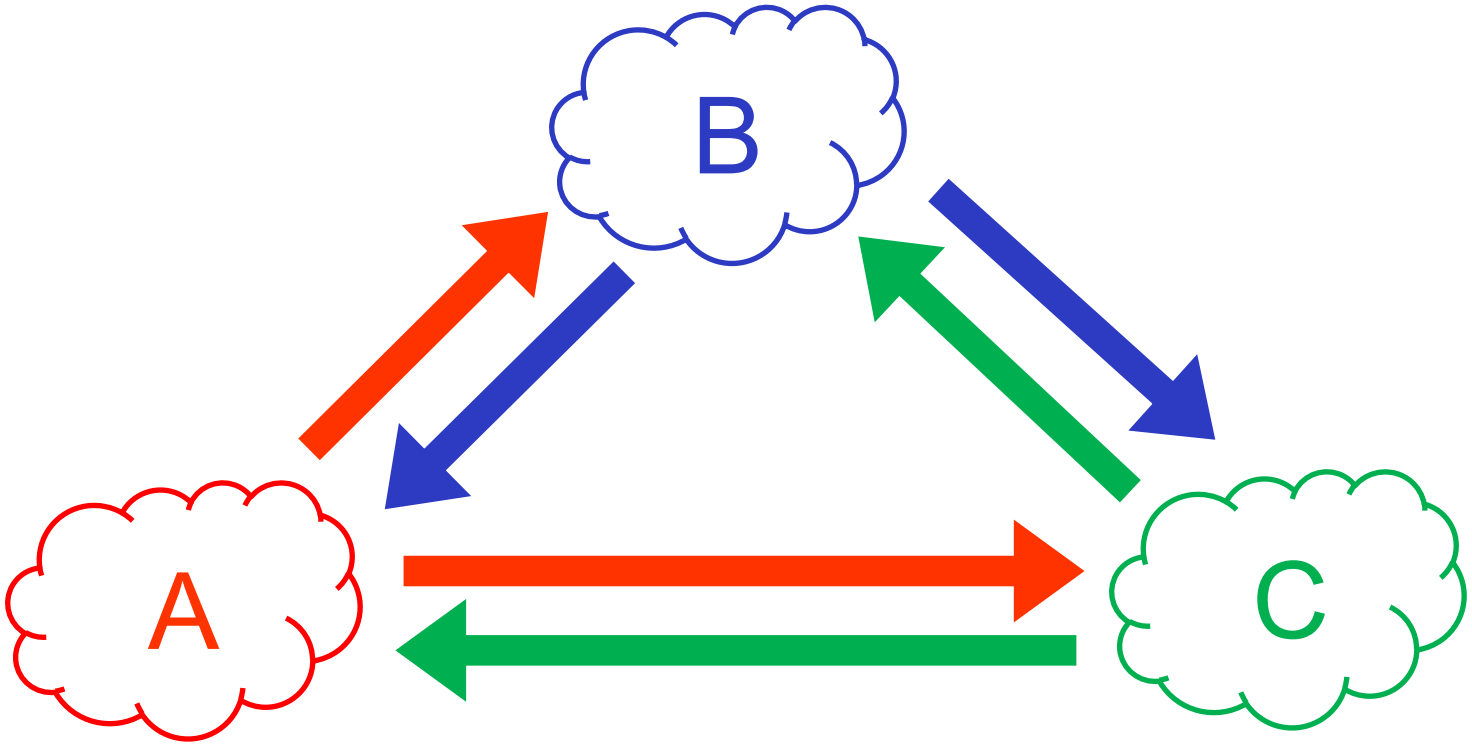
Access Token Reuse

- Client accesses resource server it was not granted access to
- Resource Server reuses client's access token at another resource server
- Solutions:
  - 1) Audience restricted access token
  - 2) Sender constrained access token

Audience Restricted Access Token



Parties are both client and resource server



Sender constrained access token

Eg: UTM Security Model

- UTM: UAS Traffic Management
- UAS: Unmanned Aircraft System (drones)
- Aviation authority is Authorization Server and determines scopes for each party
- Each party may call any other party
- One access token per client simpler for AS

- **Server constricted access tokens**
- **NOT COVERED IN CURRENT DRAFT**

HTTP 401 response

- Client discovers Authorization Server
- Client discovers resource URI

```
HTTP/1.1 401 Unauthorized
    WWW-Authenticate: Bearer ...
    Link: <https://api.example.com/resource">;
       rel="resource_uri",
 <https://as.example.com/.well-known/oauth-authorization-server>;
       rel="oauth_server_metadata_uri”
```

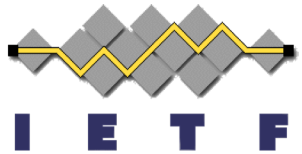
- Client confirms resource URI in host and path

Access Token Request

- Client includes resource URI in request

grant_type=client_credentials
&scope=example_scope
&resource=**https**%3A%2F%2F**api.example.com**%2F**resource**

Access Token Includes Resource URI



- If JWT, “aud” includes resource URI
- Resource server checks resource URI is in access token

Discussion

- URI for resource?
- “Link” header
  - **`“resource_uri”`**
  - **`“oauth_server_metadata_uri”`**
- Support multiple resources in access token request?
- Client PoP mechanisms?

Next Steps

- Add resource URI to code flow
- Sender constrained access tokens?
- OAuth WG adoption?