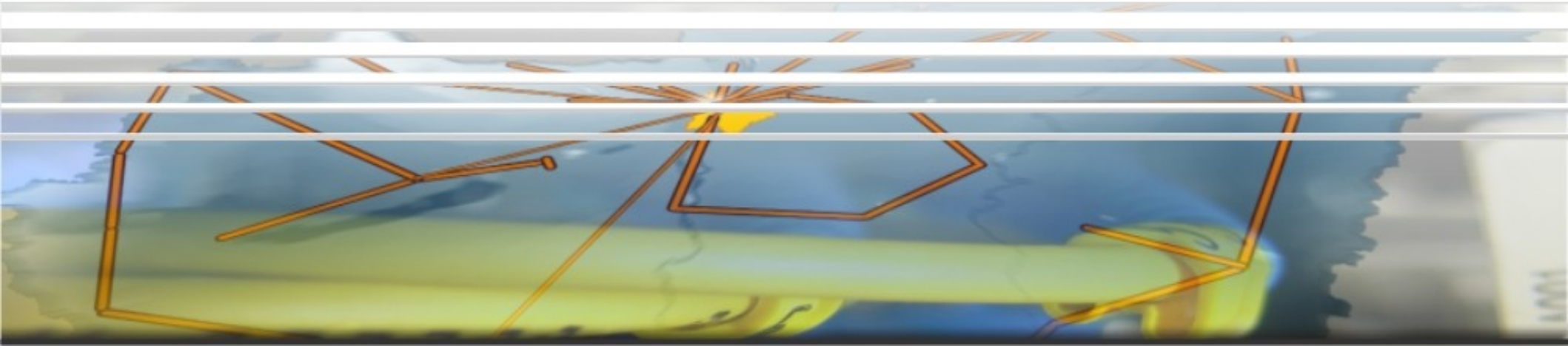
STUN/TURN federation benefits in the WebRTC world



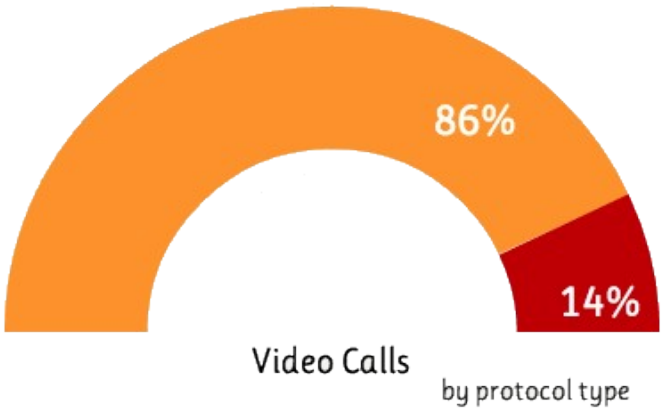
10/27/15
Stockholm / Sweden

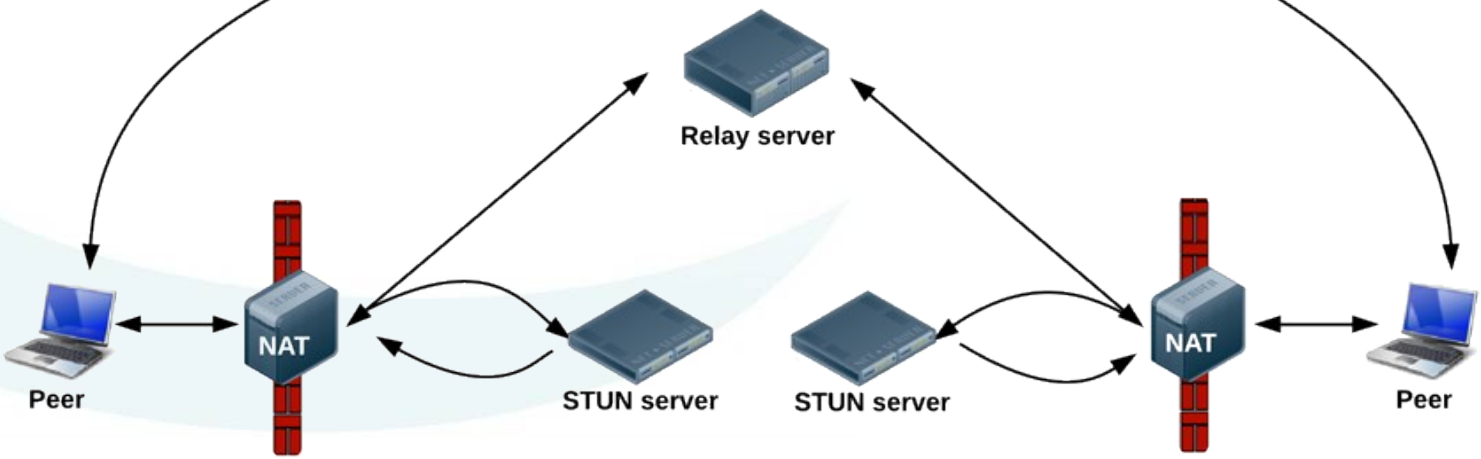
Mihály Mészáros

Quick Survey

- Protocols
  - How many of you familiar with (Symmetric) NAT?
  - How many of you familiar with STUN?
    - STUN (Session Traversal Utilities for NAT)
  - How many of you familiar with TURN?
    - TURN (Traversal Using Relays around NAT)
  - How many of you familiar with ICE?
- STUN/TURN Service
  - How many of you provide STUN service for your community?
  - How many of you provide TURN service for your community?

ICE (TURN/STUN) Brief overview





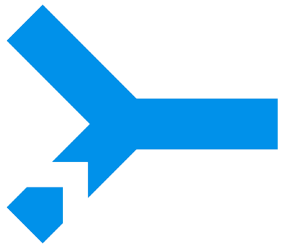
ICE RFC5245 (STUN/TURN)

- Steps
  - Candidate gathering
    - STUN (detect reflexive)
    - TURN (allocation)
  - Prioritisation
  - Exchange
  - Connectivity checks
    - Prune duplicates
  - Coordination
    - Controller,controlled
    - Nomination (aggressive,regular)
    - Keep-alive checks
  - Communication

- Goals
  - Find the best path
  - Firewall traversal
  - IPv4, IPv6 Inter-working
  - Ready to handle Multiple IP address

IP address leakage

- The issue
  - By design browser and ICE gathers ALL(!) possible interfaces and IP address candidates to find the best way.
  - And by the way your browser expose your IP addresses
    - Private, Public, VPN etc.
- FIX is under way
  - Chrome
    - Opt-In: Network limiter Extension
    - Step2 build in the core, and make it default
  - Firefox
    - New UI tools to restrict candidates
    - Expected in FF Beta 41



Why TURN is important?

- Some cases we couldn't avoid relaying media
  - e.g.
    - Symmetric NAT,
    - Strict ALG,
    - etc.
  - Even ICE-TCP + media bundling + rtp/rtcp multiplex on 443 port could fail with a strict Firewall

- First or Last?
  - Philipp Hancke pointed out an interesting fact:
    - Many WebRTC implementations go for relay first, and this way not use TURN as the last resort, but the relay is used firstly to reduce call establishment/setup time.
    - After established call switch to direct P2P

Short Term & Long Term

- STUN RFC5389 defines two Authentication mechanism
  - Short-term Credential mechanism
    - Time limited, e.g. applicable for a session
    - Mainly used for ICE connection checks
  - Long-term Credential mechanism
    - Not time limited
    - Mainly used for
      - STUN reflexive address detection client auth
      - TURN relay address allocation client auth

Why we need a new Time limited Long Term Credential auth concept?

- Why Long Term Credential is not suitable?
  - The problems with the TURN long-term auth exchange are documented in draft-reddy-behave-turn-auth
    - TURN password must be kept secret (hard for WebRTC apps)
    - TURN password vulnerable to offline dictionary attacks on MESSAGE-INTEGRITY
    - TURN server must consult a password database to verify MESSAGE-INTEGRITY
    - TURN username value is passed in the clear, can be used for traffic analysis
- Why Short Term Credential is not suitable?
  - STUN defines a short-term credential mechanism, but this mechanism doesn't support nonces, opening the door for trivial replay attacks

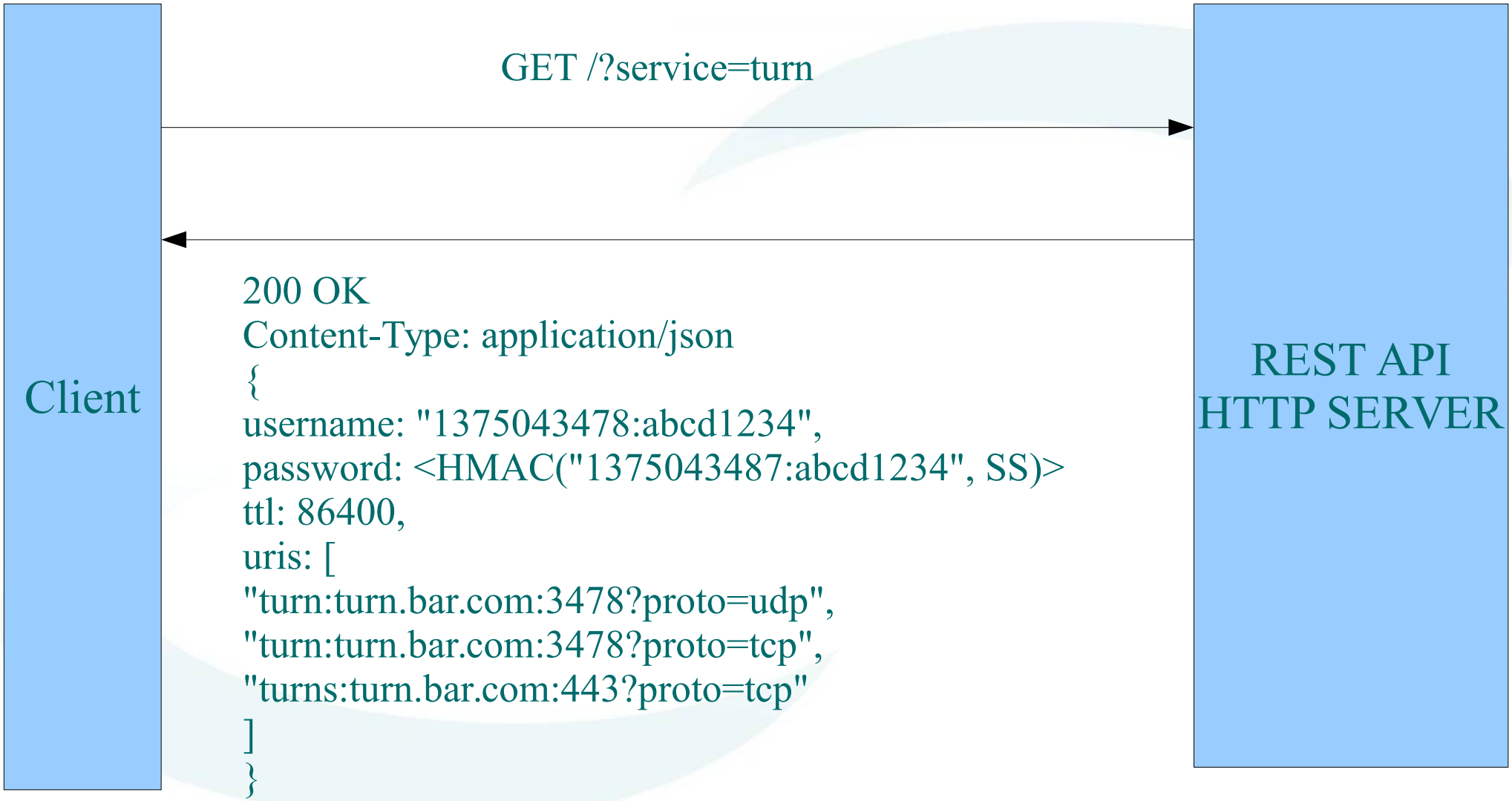
TURN REST API
Improved Stateless Authentication

- Time limited Long Term Credential
- Proposed solution:

  Client makes a HTTP request to a web service to get ephemeral (time-limited) credentials:

  - No long-term credentials to keep secret; even if discovered, credential usefulness is limited
  - Username contains no externally-identifying information
  - Password is machine-generated, to prevent dictionary attacks
  - Response also includes location of TURN server, avoiding complex SRV lookups

TURN REST API
Get Credential



TURN REST API
Credential Verification

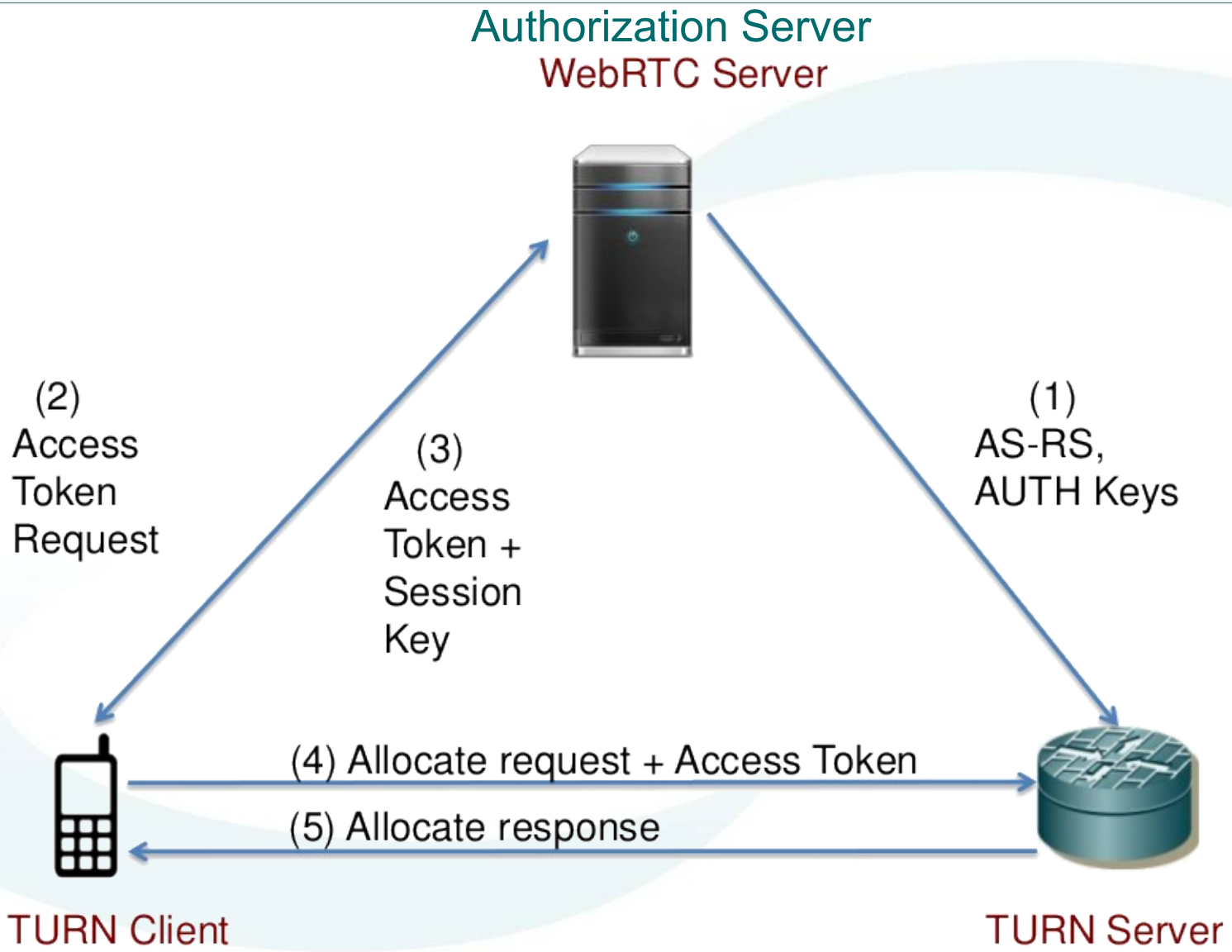
- Credential Verification
  - While the TURN server could verify credentials against the HTTP server, the draft suggests a stateless design that requires no backchannel.
  - Username is credential expiration timestamp + any desired application data
    e.g. "1375043478:abcd1234"
  - Password is HMAC(username, SS), where SS is a shared secret key between HTTP and TURN servers
    e.g. <HMAC("1375043487:abcd1234", SS)>
  - To get HA1, TURN server simply does
    MD5(<username>:<realm>:<hmac>)

TURN REST API (STUN Request & Verify)

- ALLOCATE REQUEST
  - USERNAME: 1375043478:abcd1234
  - REALM: bar.com
  - NONCE: abcd1234
  - MESSAGE-INTEGRITY:
    HMAC(M, MD5("1375043478:abcd1234:bar.com:<hmac-password>"))

- Stateless TURN Auth: Verify
  - Parse timestamp from USERNAME (1375043478)
  - Check that timestamp is in the future
  - Compute password: HMAC(1375043478:abcd1234, SS)
  - Compute HA1: MD5(1375043478:abcd1234:bar.com:<hmac-password>)
  - MESSAGE-INTEGRITY verify against HMAC(M, HA1)
  - If it's cool, return success response
  - No communication with HTTP server needed!

3rd Party auth for TURN using OAuth



Federated STUN/TURN Service
GÉANT STUN/TURN Service

STUN/TURN Federation service usage

- STUN/TURN Server potential users
  - SIP User Agents
    - VoIP
      - Soft/Hard phones
    - Telepresence / VideoConference
      - Soft/Room VC systems
    - Long Term Credential auth mechanism
  - XMPP/Jabber/Jingle/COLIBRI Clients.
    - Long Term Credential auth mechanism
  - Web Applications (WebRTC)
    - Time limited Long Term Credential (REST API)
    - OAuth token/assertion auth

Federated Standard auth service

- Proposed architecture for SIP/XMPP, whatever any IETF standard based STUN Long-Term Credential without time limit.
  (realm, username, password)
  - round-robin DNS based load-balancing
  - Most terminal implementations only supports this mechanism
  - End User Interaction:
    - Add a from on a GÉANT (eduGAIN authenticated) web site where the authenticated user could request access to this service with a single click.
    - During the registration process from the email/eppn attribute we could extract the username and realm/domain, and use it for turn username and realm/domain.

Federated Time limited Long-Term Credential (REST API) For Web Applications

- Proposed architecture for WebRTC or other web users for WebRTC using REST API
  - An NREN service operator who runs a WebRTC Service, can register his service on the GÉANT STUN/TURN service website after AAI auth.
  - It will receive a secret key that is unique per service.
  - With this shared unique service key, the service could get access to a time limited shared secret key that the turn servers are using.
  - The Web Application backend use this key to send out a simple REST API calls and get a time limited TURN credential and turn server services and IP address/URIs.
  - Based on the API call source IP and/or from the operational turn servers load, we could provide the best/nearest turn server IP to the user. (Load-balancing and Nearest available)

STUN/TURN Implementation

- CoTURN
  - OpenSource (https://github.com/coturn/coturn)
  - The freshest TURN standard implementation
    - Very reliable implementation.
  - Widely used by WebRTC market players
    - Even the biggest player like Google use this implementation.
  - Most important Features:
    - Support STUN (booth Classic and New)
    - TURN client support (TCP, UDP, TLS, DTLS, SCTP)
    - TURN relay support (TCP/UDP)
    - Supports multiple backend databases (SQLite, MySQL, PostgreSQL, Redis, MongoDB)
    - Auth (classic, REST API, Oauth)
    - Both IPv6/IPv4

CookBook to pilot such federated service

- Deploy a host with IPv4,IPv6 public address, (and certificate for TLS)
- Configured CoTurn daemon
  - Standard Auth
    - Distributed realm, user, password database
    - Develop a Web interface for managing these Long Term Credentials.
  - REST API
    - Develop a shared secret authenticated simple REST API, to get time limited long term credential for the backend of web applications, that could be used as auth credential with STUN/TRUN severs.
    - Develop an eduGAIN AAI protected web registry for service providers. Register a service to receive a generated shared key that is used to get access to the REST API..
    - Setup a database and use as backend to store these shared secrets

Summary

- Benefits for the community
  - Better firewall traversal experience for End user.
  - Smooth IPv6 transition for the end users
  - IETF standard based firewall traversal instead of tunnels
  - Reliable distributed STUN service for GÉANT community services
    - For reflexive address detection
  - Reliable distributed TURN service for GÉANT community services
    - For media relaying
- It would be beneficial for wide range of applications and services that use or will use Standard based firewall traversal technologies. e.g.:
  - standard auth for SIP(VoIP,VC),XMPP(jingle)
  - REST API, Oauth for WebApp-s (WebRTC)

References

- http://sdstrowes.co.uk/talks/20081031-ice-turn-stun-tutorial.pdf
- http://www.slideshare.net/webrtclive/kranky-geek-google-team
- https://youtu.be/Gr7PJAyMJdU
- http://tools.ietf.org/html/rfc5389#section-10.2
- https://tools.ietf.org/html/draft-reddy-behave-turn-auth
- http://tools.ietf.org/html/draft-uberti-behave-turn-rest-00
- https://www.ietf.org/proceedings/87/slides/slides-87-behave-10.pdf
- https://tools.ietf.org/html/rfc7635

  https://www.ietf.org/proceedings/90/slides/slides-90-tram-6.pdf
- https://groups.google.com/forum/#!topic/mozilla.dev.media/L6Rx9ubSjMc
- https://github.com/coturn/coturn
- https://github.com/coturn/coturn/wiki/turnserver
- https://nws.niif.hu/ncd2012/docs/phu/054.pdf
- https://rfc5766-turn-server.googlecode.com/svn-history/r217/trunk/docs/TURNServerRESTAPI.pdf

Thank You!