ALCATEL-LUCENT RAINBOW™

Network Requirements

GETTING STARTED GUIDE Ed 6

APRIL 2017

Author: R&D - Cloud Services

Disclaimer

This documentation is provided for reference purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this documentation, this documentation is provided “as is” without any warranty whatsoever and to the maximum extent permitted.

In the interest of continued product development, ALE International reserves the right to make improvements to this document and the products it describes at any time without notice or obligation.

Copyright

Contents

Glossary

ALE: Alcatel-Lucent Enterprise

PBX: Private Branch Exchange

HTTP: Hyper Text Transfer Protocol

HTTPS: Hyper Text Transfer Protocol Secured

ICE: Interactive Connectivity Establishment – RFC 5245

STUN: Simple Traversal of UDP through NAT – RFC 5389

TURN: Traversal Using Relays around NAT – RFC 5766

DTLS-SRTP: Datagram Transport Layer Security – Secured Real Time Protocol

1 Introduction

This guide provides technical requirements to connect Rainbow clients and Agent to Rainbow Cloud services.

2 Overview

Alcatel-Lucent Enterprise (ALE) is introducing Alcatel-Lucent Rainbow, an overlay cloud service operated by ALE. Rainbow offers contact management, presence, persistent messaging, audio/video, screen and file sharing, with PSTN termination and API openness to integrate with existing customer PBXs, machines and apps.

Rainbow's clients and agents connect to Rainbow cloud services using Web protocols.

More details about Web protocols used are provided in this document.

3 History

Modifications	Date	Edition
HTTP vs. HTTPS cleanup	04/04/2017	Ed 05
Minor change (legacy PBX Agent removed)	31/03/2017	Ed 04
Information on bandwidth added (chapter 5.5)	08/03/2017	Ed 03
Chapter 5.7.3 added, Chapter 6.1 modified	05/01/2017	Ed 02
Creation of document	27/10/2016	Ed 01

4 Related documents

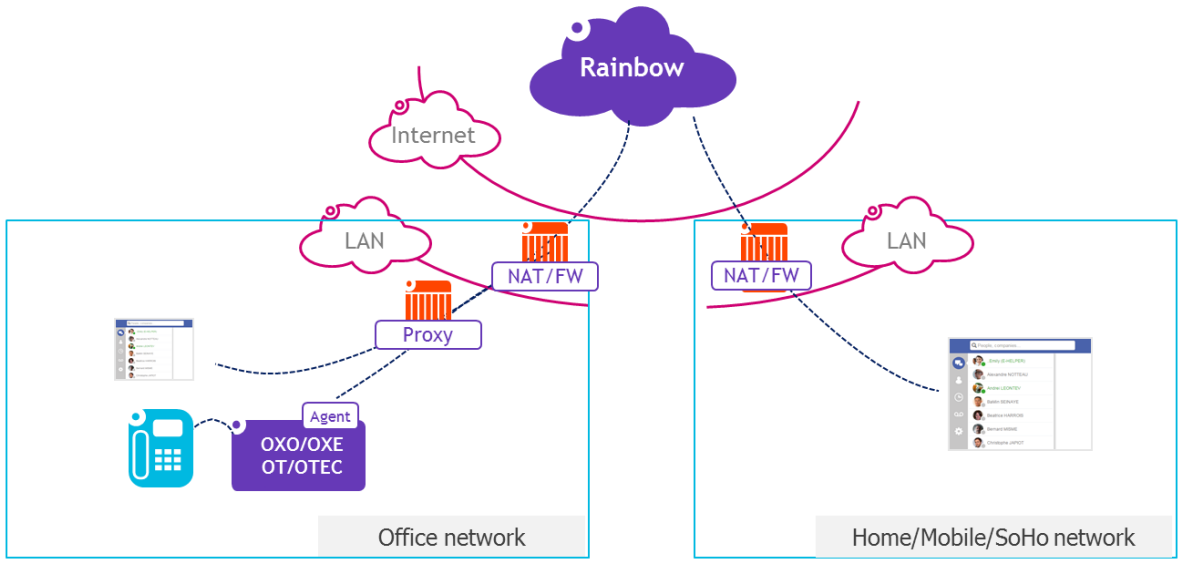
None

5 Requirements

5.1 Global Overview

The following picture provides the global overview of Rainbow from network perspective :

Rainbow global picture



5.2 Used Protocols

The Rainbow solution provides multiple client-side applications to connect to the service:

- A Web-based Qt-contained Desktop application for Windows and OSX.
- A Web application for Chrome/Firefox browsers.
- An iOS native application.
- An Android native application.
- An Agent to connect the PBX (can be integrated with the PBX)

All applications aim at providing the same level of services and features and interact with server-side components the following way:

- Through HTTPS (443) for all REST API communications and resources loading.
- Through secure Web Sockets (WSS, 443) for all XMPP messages and notifications.
- Through DTLS and SRTP with or without STUN/TURN for WebRTC-based audio/video media streams.

5.3 Connections and Ports used

Depending on the presence of an HTTP Proxy (Web Proxy), following connections take place between Rainbow client/Agent and Rainbow Cloud Services

Protocols	no Proxy, or Proxy not used	HTTP Proxy configured and used
HTTPS (Resources and REST API)	Direct, TCP port 443	Enforced through the Proxy, TCP port 80/443
Web Sockets (XMPP)	Direct, TCP port 443	WebSocket, enforced through the Proxy, via HTTP switch, TCP port 443
ICE/TURN(s)	Direct, TCP port 80 and 443	Enforced through the proxy via HTTP Connect tunnel TCP port 80 or 443
DTLS-SRTP (Media)	Direct, TCP or UDP, dynamic port range (49152 – 65535) for local call, TCP/port 80/443 for Wan if turn is used	Encapsulated in TURN(s), enforced through the proxy via HTTP Connect tunnel TCP port 80/443

5.4 Domain used

Rainbow cloud services use the following domains:

Used by	Purpose	Domains
Rainbow Clients	Resources (website, images, client package, Agent package, …)	web.openrainbow.com
Rainbow Clients	REST API	openrainbow.com
Rainbow Clients	XMPP over Secured WebSockets	openrainbow.com
Rainbow Clients	STUN/TURN	turn-eu1.openrainbow.com, turn-na1.openrainbow.com, turn-as1.openrainbow.com, turn-oc1.openrainbow.com

5.5 Bandwidth requirement

5.5.1 WebRTC

Rainbow WebRTC communication currently rely on the following codecs:

- opus for audio
- vp8 or H.264 for Video and Screen Sharing,

WebRTC codecs are able to dynamically throttle their bitrates, depending on network performance observed. The following table provides bandwidth requirement per media :

Media	Nominal bandwidth	Lowest Bandwidth	Comment
Audio	80kbps	15kbps	
Video	350kbps	150kbps	
Screen Sharing	15-400kbps		Depends if shared content is dynamic

5.6 Configuration of border elements in enterprise

To allow Rainbow to operate properly, border elements like DNS, HTTP Proxy or Firewall must be configured to allow accessing domains and protocols listed in the table chapter 5.3 and 5.4.

Regarding range of ports, if you are in one of the two following cases, there is no specific port range to configure in border elements:

If an HTTP Proxy is configured, all the traffic will be directed through the proxy (HTTP Connect method),

If no HTTP Proxy is configured, that probably means that any direct traffic over arbitrary TCP/UDP ports is allowed.

If you are in the second case, but traffic is filtered by a border element, please, check chapter 5.3 and 5.4.

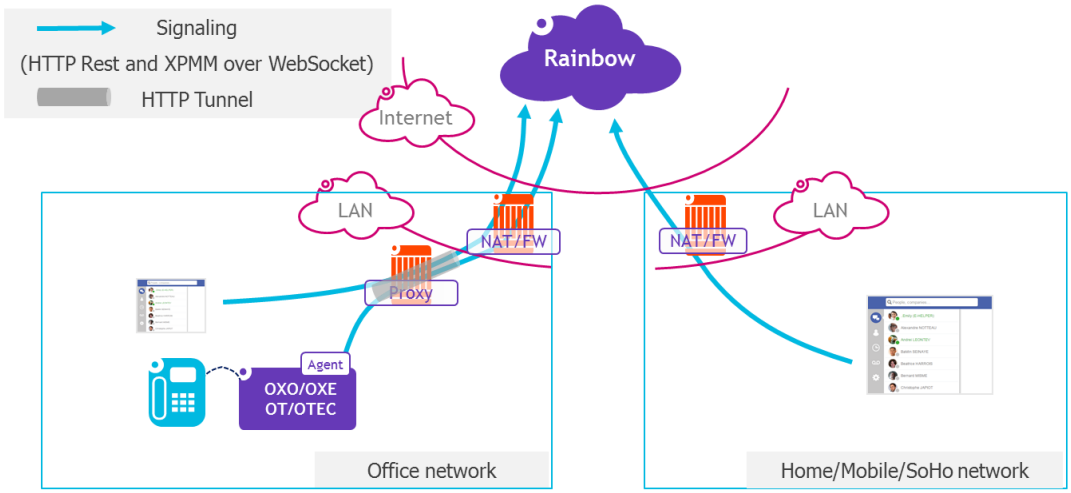
5.7 Connections illustrated

5.7.1 Signaling

For signaling, HTTPS/REST and Secured WebSockets protocols are used.

If a HTTP Proxy is configured, HTTP Proxy is used. In such case, HTTP Proxy must support Secured WebSocket (HTTP Upgrade to switch to wss protocol).

Signaling



5.7.2 WebRTC/Media

ICE (Internet Connectivity Establishment) procedure and STUN/TURN protocols are used to dynamically determine how the media will be routed between two Rainbow clients.

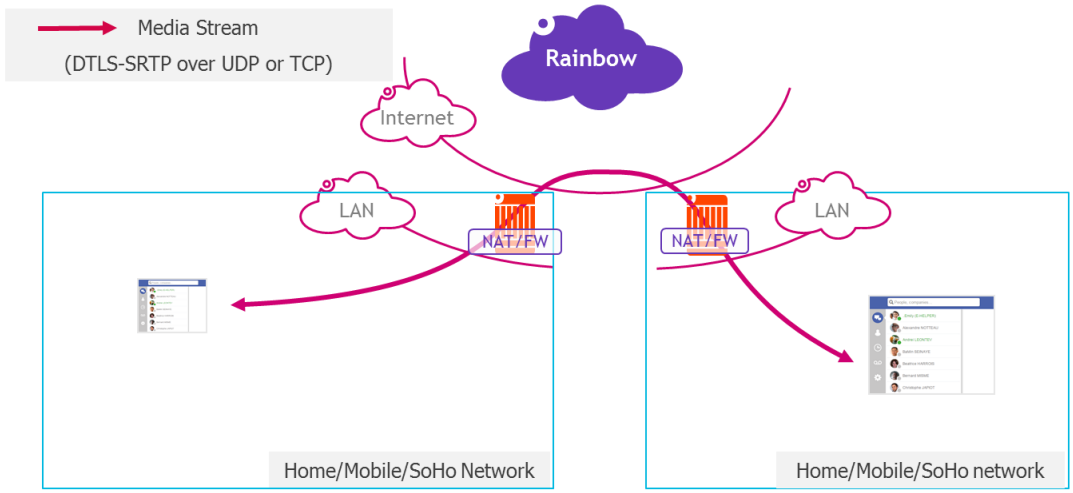
Basically, when a WebRTC communication takes place, client proceeds to the following steps:

- For each client, gather candidates addresses (A candidate is a transport address, a combination of IP address and port for a particular transport protocol, allocated on local interface and on TURN server in case of TURN is required, local interface are for example wired Ethernet interface or WiFi interface for a PC),
- Exchange candidates with the peer client,
- Check connectivity for candidates between both clients and select a working peer of candidates.

5.6.2.1 Exemple 1 : P2P WebRTC (Basic network)

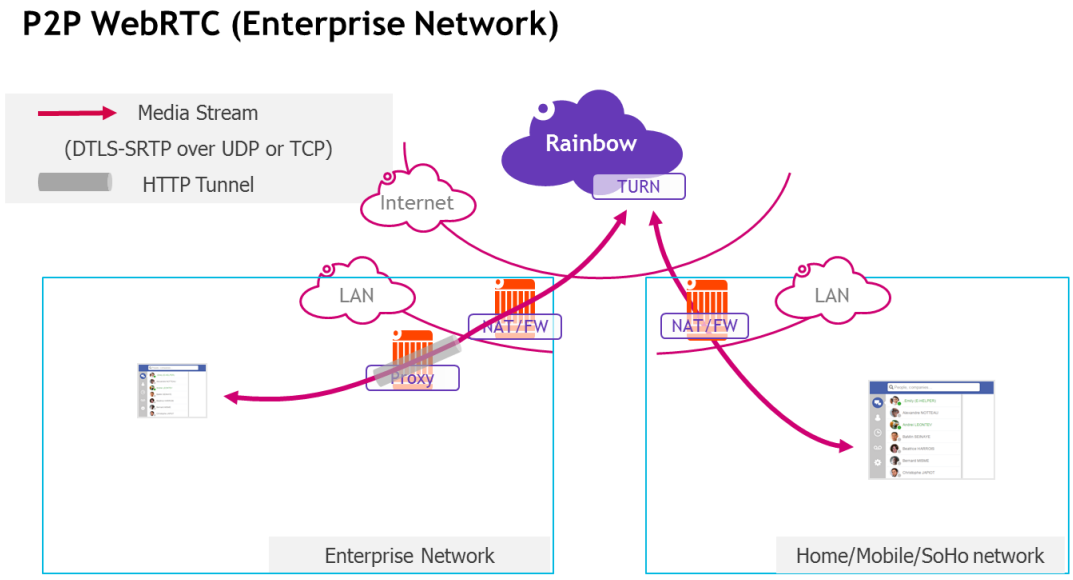
This is the simplest case. When network elements between the two parties are simple NAT/FW (like home router), in most of the time, WebRTC is able to establish a direct media path (UDP or TCP).

P2P WebRTC (SoHo/Home network)



5.6.2.1 Exemple 2 : P2P WebRTC (Enterprise network)

When it comes to enterprise network, most likely an HTTP Proxy will be there. In such case, HTTP Tunnel and TURN are used to establish the media path.

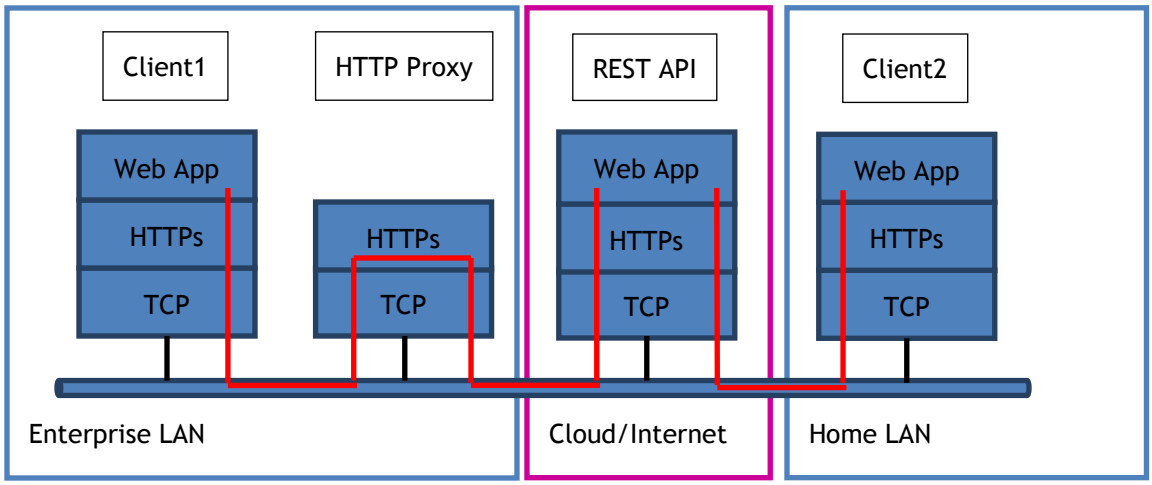


Note: to simplify figures, only one TURN server is illustrated. For a P2P communication, depending on geography and network performance, up to two TURN server could be used to establish a communication.

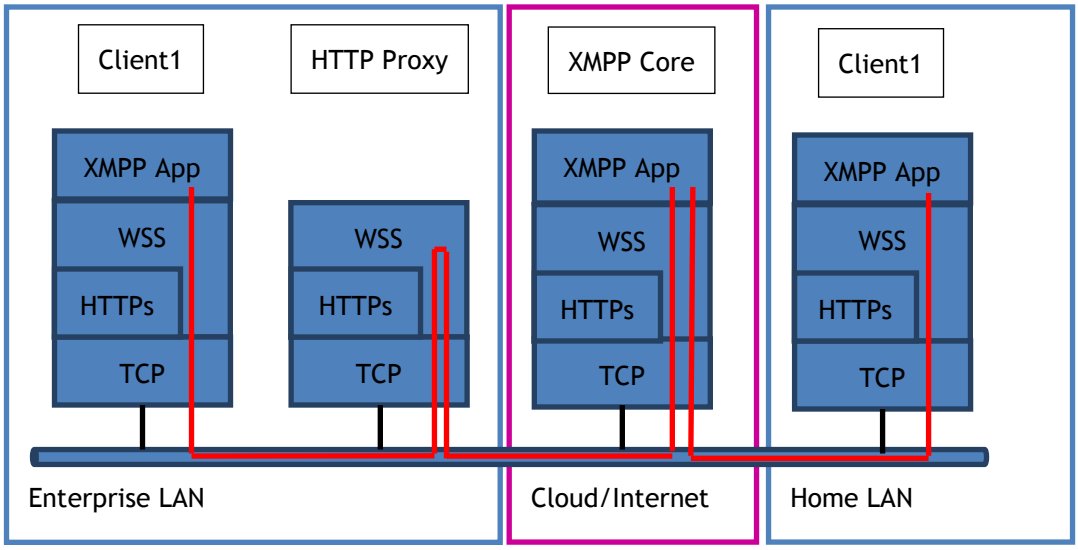
5.7.3 Detailed call-flow of HTTPS/REST, XMPP and ICE connections

The following figures illustrate a case where Rainbow Client1 make an Audio/Video call to Rainbow Client2. Rainbow Client1 is in an enterprise environment with NAT/FW and HTTP Proxy border elements. Rainbow Client2 is in a Home network with simple NAT/FW as border element (home router/box).

Network layers (Rest API):

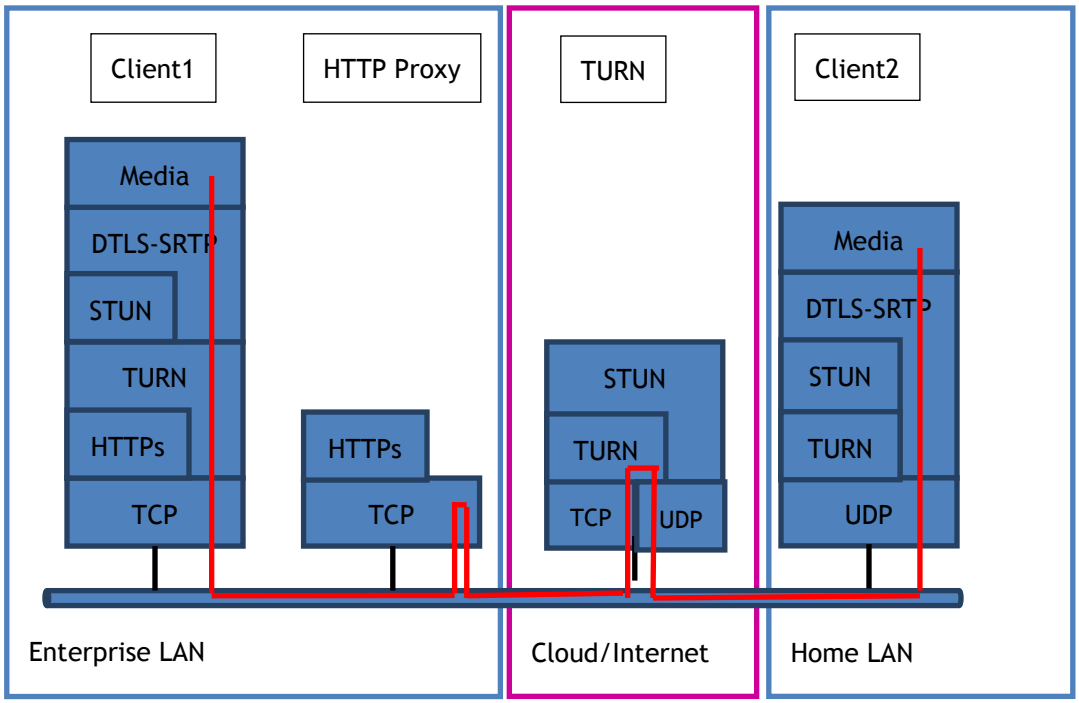


Network layers (XMPP):



HTTPs is used to setup WSS protocol (RFC6455)

Network layers (Media):



HTTPs is used to setup HTTP Tunnel to TURN server

STUN is a protocol helper to check connectivity

TURN is a protocol helper to pass-through NAT/FW/HTTP Proxy

Call Flow:

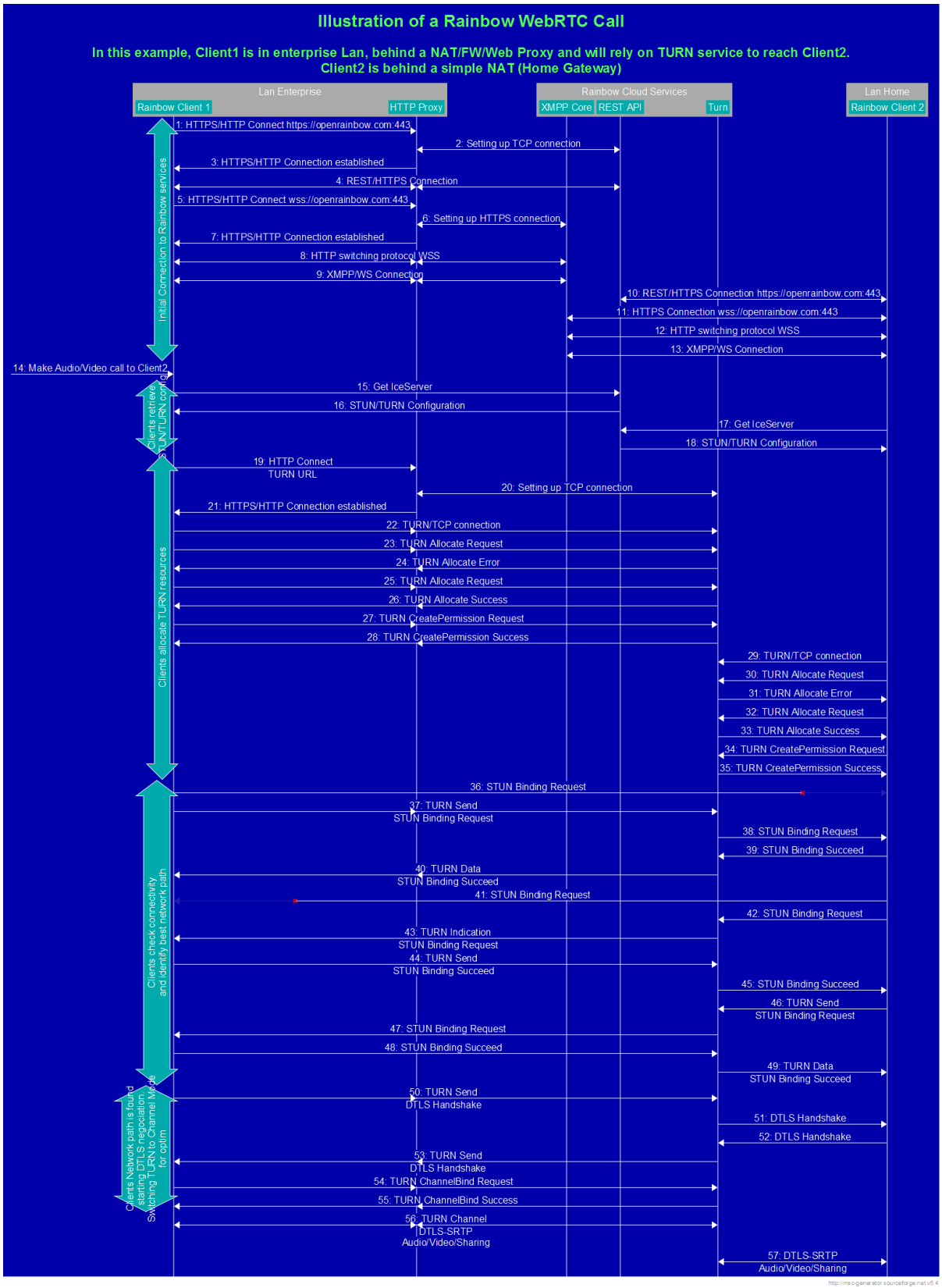


Illustration of a Rainbow WebRTC Call
In this example, Client1 is in enterprise Lan, behind a NAT/FW/Web Proxy and will rely on TURN service to reach Client2.
Client2 is behind a simple NAT (Home Gateway)
Lan Enterprise
Rainbow Client 1
HTTP Proxy
Rainbow Cloud Services
XMPP Core
REST API
Turn
Lan Home
Rainbow Client 2
Initial Connection to Rainbow services
1: HTTPS/HTTP Connect https://openrainbow.com:443
2: Setting up TCP connection
3: HTTPS/HTTP Connection established
4: REST/HTTPS Connection
5: HTTPS/HTTP Connect wss://openrainbow.com:443
6: Setting up HTTPS connection
7: HTTPS/HTTP Connection established
8: HTTP switching protocol WSS
9: XMPP/WS Connection
10: REST/HTTPS Connection https://openrainbow.com:443
11: HTTPS Connection wss://openrainbow.com:443
12: HTTP switching protocol WSS
13: XMPP/WS Connection
14: Make Audio/Video call to Client2
Clients retrieve STUN/TURN config
15: Get IceServer
16: STUN/TURN Configuration
17: Get IceServer
18: STUN/TURN Configuration
Clients allocate TURN resources
19: HTTP Connect
TURN URL
20: Setting up TCP connection
21: HTTPS/HTTP Connection established
22: TURN/TCP connection
23: TURN Allocate Request
24: TURN Allocate Error
25: TURN Allocate Request
26: TURN Allocate Success
27: TURN CreatePermission Request
28: TURN CreatePermission Success
29: TURN/TCP connection
30: TURN Allocate Request
31: TURN Allocate Error
32: TURN Allocate Request
33: TURN Allocate Success
34: TURN CreatePermission Request
35: TURN CreatePermission Success
Clients check connectivity and identify best network path
36: STUN Binding Request
37: TURN Send
STUN Binding Request
38: STUN Binding Request
39: STUN Binding Succeed
40: TURN Data
STUN Binding Succeed
41: STUN Binding Request
42: STUN Binding Request
43: TURN Indication
STUN Binding Request
44: TURN Send
STUN Binding Succeed
45: STUN Binding Succeed
46: TURN Send
STUN Binding Request
47: STUN Binding Request
48: STUN Binding Succeed
49: TURN Data
STUN Binding Succeed
Clients Network path is found starting DTLS negociation Switching TURN to Channel Mode for optim
50: TURN Send
DTLS Handshake
51: DTLS Handshake
52: DTLS Handshake
53: TURN Send
DTLS Handshake
54: TURN ChannelBind Request
55: TURN ChannelBind Success
56: TURN Channel
DTLS-SRTP
Audio/Video/Sharing
57: DTLS-SRTP
Audio/Video/Sharing

- Steps 1-9 : Clients1, behind HTTP Proxy, establish HTTPs sockets and Secured WebSockets(For XMPP) through the HTTP Proxy.
- Steps 10-13 : Clients2, behind simple NAT, establish regular HTTPs sockets and Secured WebSockets (For XMPP) directly to Rainbow Cloud Services.
- Step 14 : Client1 make an Audio/Video call to Client2
- Steps 15-18 : Client1 and Client2 retrieves ICE configuration (list of TURN servers)
- Steps 19-28 : Client1 allocates TURN resources (through HTTP Proxy)
- Steps 29-35 : Client2 allocates TURN resources (direct connection to TURN)
- Steps 36-49 : Client1 and Client2 perform STUN connectivity check for various options (in this example, we illustrate the case where direct STUN connectivity check would failed)
- Steps 50-53 : Client1 and Client2 have found a network path through the TURN server. They start to initiate the DTLS-SRTP handshake.
- Steps 54-55 : Client1 ask TURN to switch to Channel mode to optimize network bandwidth (reduce TURN header overhead).
- Steps 56-57 : DTLS-SRTP is established, Audio/Video media starts to flow between Client1 and Client2

6 Limitations, Restrictions and workarounds

6.1 HTTP Proxy

If an HTTP Proxy is configured and/or selected at OS level (fixed configuration, auto-detect, script, …), by design, Browsers and Rainbow desktop client as well will always rely on this HTTP Proxy to reach Rainbow cloud services (for all protocols used, including HTTPS/REST, XMPP over Secured WebSockets and TURN). It could append that the proxy in place in enterprise is not able or is not willing to pass-through some protocols (TURN for example). Capability and willingness of the HTTP Proxy must be checked by IS/IT team in case of trouble to start Rainbow clients or to establish Audio/Video call between Enterprise Lan and external users.

End of Document