**WebRTC**

[[Kartik Tomar](https://kartik-tomar.medium.com/?source=post_page-----35623cfbb6d0--------------------------------)](https://kartik-tomar.medium.com/?source=post_page-----35623cfbb6d0--------------------------------)

[Kartik Tomar](https://kartik-tomar.medium.com/?source=post_page-----35623cfbb6d0--------------------------------)

·

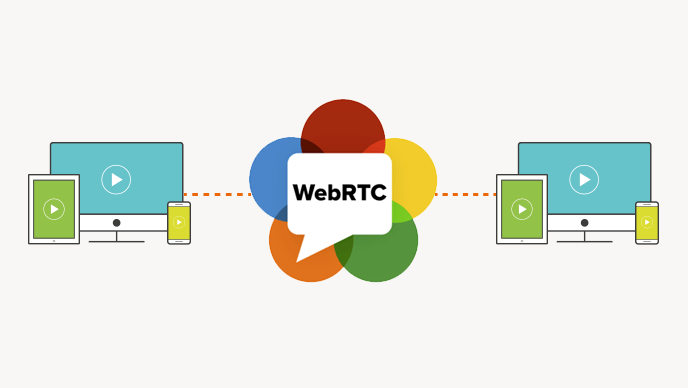
[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F7703ba92134f&operation=register&redirect=https%3A%2F%2Fkartik-tomar.medium.com%2Fwebrtc-35623cfbb6d0&user=Kartik+Tomar&userId=7703ba92134f&source=post_page-7703ba92134f----35623cfbb6d0---------------------post_header-----------)

8 min read

·

Aug 21, 2023

6



WebRTC

**Title: The Digital Odyssey: Alice, Bob, and the Tale of WebRTC**

Once upon a digital era, in the bustling city of Browserland, Alice wanted to start a video chat with her good friend, Bob. But how would this complex task unfold in the vast realm of the internet? Our tale revolves around this very mission, exploring the magics of WebRTC, NAT, and SDP.

The technologies behind WebRTC are implemented as an open web standard and available as regular JavaScript APIs in all major browsers. For native clients, like Android and iOS applications, a library is available that provides the same functionality. The WebRTC project is open-source and supported by Apple, Google, Microsoft and Mozilla, amongst others.  
Unlike all other browser communication, WebRTC transports its data over UDP (User Datagram Protocol). However, it might just not be UDP in some cases TCP too.  
WebRTC is mainly used for video communications and Audio and Video Engines are an integral part of that but even before that we need to establish connections between peers (browsers). And we will discuss ways and challenges one might face in doing so.

**Chapter 1: The Invitation**

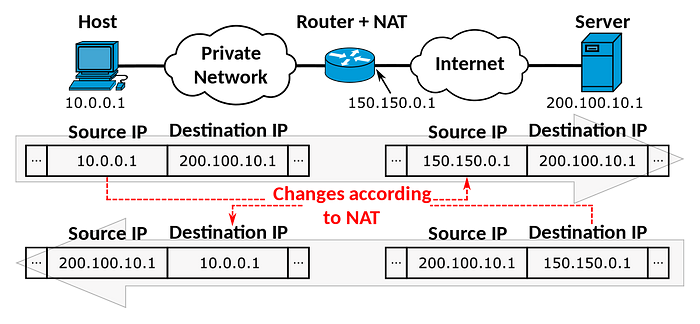
Alice opened her browser, eager to chat with Bob. Little did she know that her browser had powerful tools hidden beneath its surface, ready to assist. One such tool was **WebRTC**, a collection of protocols and magic spells (or so Alice thought) that allowed browsers like hers to communicate in real-time.

**Chapter 2: The Many Gates of NAT Kingdom**

To reach Bob, Alice first had to traverse the challenging terrain of the **NAT Kingdom**. Here, guardians known as Network Address Translations protected the city gates. Their role? To ensure many citizens of Browserland could share one public address to converse with the outer world.

To understand Web RTC connection we first need to understand NAT first. To access the Internet, one public IP address is needed. The idea of NAT is to allow multiple devices to access the Internet through a single public address. To achieve this, the translation of a **private IP** address to a **public IP** address is required. **Network Address Translation (NAT)** is a process in which one or more local IP address is translated into one or more Global IP address and vice versa in order to provide Internet access to the local hosts. Also, it does the translation of port numbers i.e. masks the port number of the host with another port number, in the packet that will be routed to the destination. It then makes the corresponding entries of IP address and port number in the NAT table. NAT generally operates on a router or firewall.

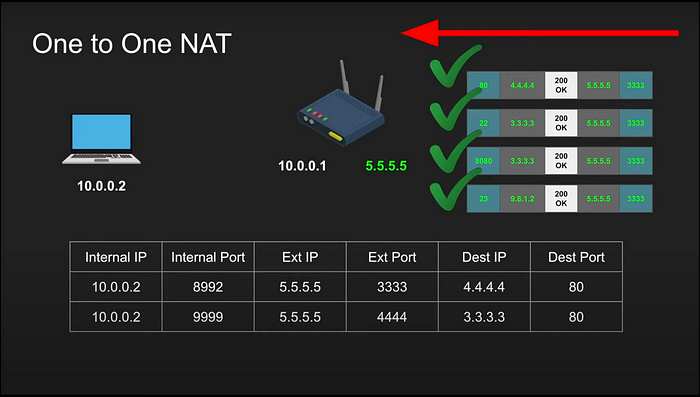
Generally, the border router is configured for NAT i.e. the router which has one interface in the local (inside) network and one interface in the global (outside) network. When a packet traverse outside the local (inside) network, then NAT converts that local (private) IP address to a global (public) IP address. When a packet enters the local network, the global (public) IP address is converted to a local (private) IP address.



There were various gates in NAT Kingdom:

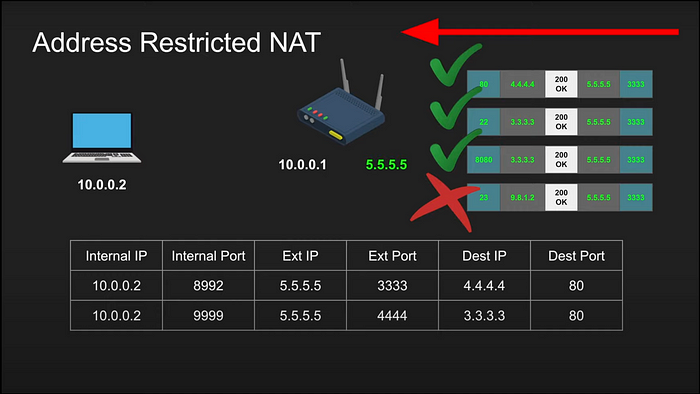
The Full-cone Gate: A friendly gate that allowed any outsider to send messages if they knew the destination.

One to One NAT (Full-cone NAT) — Also called static NAT, This means in the local area network side and outside network side we need an equal number of IP addresses to translate. One public IP is matched with one internal IP. NAT allows any packet coming to the public IP to the Internal IP mapped to it.



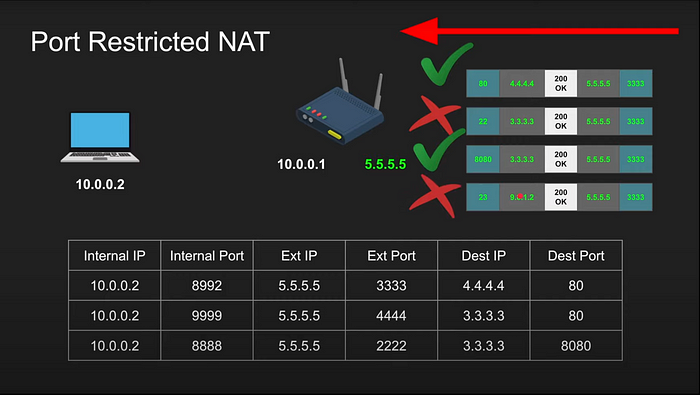
The Address Restricted Gate: A more reserved gate, allowing only outsiders that a Browserland citizen had spoken to before.

Address Resticted NAT — Packets to external IP:port on the router always maps to internal IP:port as long as source address from packet matches the table (regardless of port), Allow if we communicated with this host before



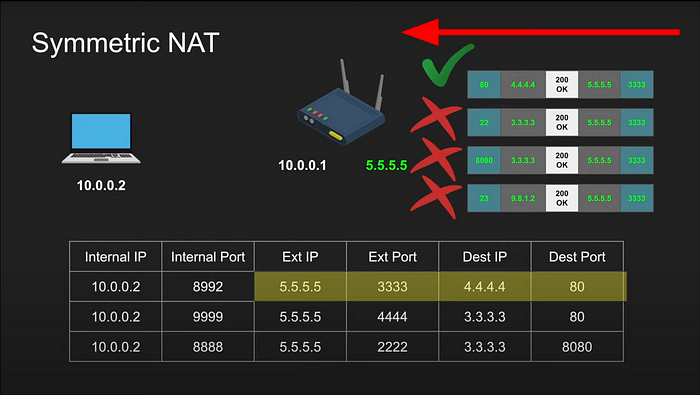
The Port Restricted Gate: Even more cautious, this gate required outsiders to use the same address and exact words (ports) the citizen had previously used.

Port Restricted NAT — Packets to external IP port on the router always maps to internal IP port as long as source address and port from packet matches the table, Allow if we communicated with this host:port before



The Symmetric Gate: The most mysterious gate of all, only allowing entry if the outsider perfectly mirrored the initial words and address used by the Browserland citizen.

Symmetric NAT- Packets to external IP port on the router always maps to internal IP port as long as source address and port from packet matches the table, Only Allow if the full pair match



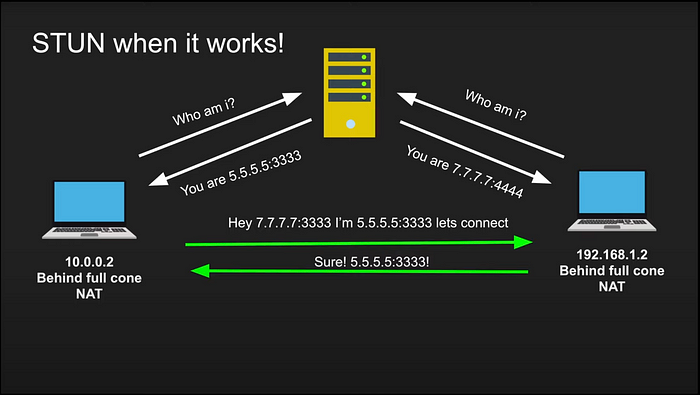
**Chapter 3: The STUNning Magician and the TURNed Tale**

To help Alice find the best path to Bob, a magician named STUN came forward. With his powers, he could reveal Alice’s public identity (IP address and port) beyond the NAT Kingdom. However, if STUN’s magic proved ineffective, there was another mystical entity known as \*\*TURN\*\*. TURN would relay Alice’s messages around the strictest of NAT gates, ensuring they reached Bob.

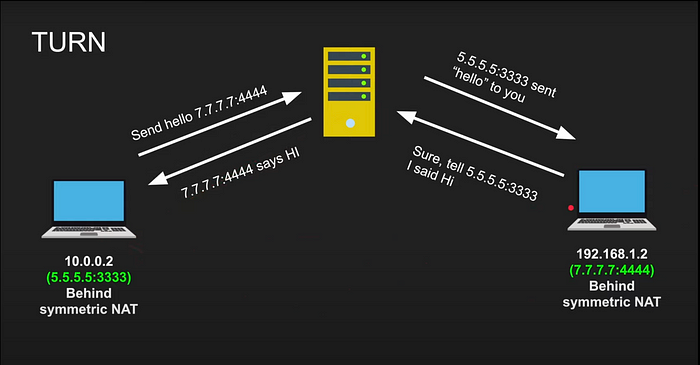
A STUN (Session Traversal of User Datagram Protocol [UDP] Through Network Address Translators [NATs]) server allows NAT clients (i.e. IP Phones behind a firewall) to set up UDP connection.

The STUN server allows clients to find out their public address, the type of NAT they are behind and the Internet side port associated by the NAT with a particular local port. This information is used to set up UDP communication between peers

* Tell me my public ip address/port through NAT
* Works for Full-cone, Port/Address restricted NAT
* Doesn’t work for symmetric NAT
* Cheap to maintain



For most WebRTC applications to function a server is required for relaying the traffic between peers, since a direct socket is often not possible, there are multiple reasons for this, one of them being that the NAT (Symmetric NAT) or firewall devices in use are not allowing such direct traffic to take place. In such cases, we route the data through an intermediary public server called TURN.



**Chapter 4: ICE and the Quest for Connection**

As Alice journeyed forward, she met a wise oracle named ICE. The oracle, with her vast knowledge, combined the powers of STUN and TURN to determine the most effective path for Alice’s messages.

**ICE** (*Interactive Connectivity Establishment*) is a framework used by WebRTC (among other technologies) for connecting two peers, regardless of network topology (usually for audio and video chat). This protocol lets two peers find and establish a connection with one another even though they may both be using Network Address Translator (NAT) to share a global IP address with other devices on their respective local networks.

* ICE collects all available candidates (local IP addresses, reflexive addresses — STUN ones and relayed addresses TURN ones)
* Called ice candidates
* All the collected addresses are then sent to the remote peer via SDP

**Chapter 5: The Scroll of SDP**

With the path determined, Alice needed a way to describe her intentions to Bob. Enter the \*\*Scroll of SDP\*\*. This ancient parchment, known formally as the Session Description Protocol, contained all the secrets — the chosen languages (codecs), rendezvous points (IP addresses), and timing information. It was a map, a blueprint, a contract between Alice and Bob.

**SDP** (Session Description Protocol) is the standard describing a peer-to-peer connection. SDP contains the codec, source address, and timing information of audio and video.

* A format that describes ice candidates, networking options, media options, security options and other stuff
* Not really a protocol its a format
* The goal is to take the SDP generated by a user and send it “somehow” to the other party

v= (protocol version number, currently only 0)  
o= (originator and session identifier : username, id, version number, network address)  
s= (session name : mandatory with at least one UTF-8-encoded character)  
i=\* (session title or short information)  
u=\* (URI of description)  
e=\* (zero or more email address with optional name of contacts)  
p=\* (zero or more phone number with optional name of contacts)  
c=\* (connection information — not required if included in all media)  
b=\* (zero or more bandwidth information lines)  
z=\* (time zone adjustments)  
k=\* (encryption key)  
a=\* (zero or more session attribute lines)  
t= (time the session is active)  
r=\* (zero or more repeat times)  
m= (media name and transport address)  
i=\* (media title or information field)  
c=\* (connection information — optional if included at session level)  
b=\* (zero or more bandwidth information lines)  
k=\* (encryption key)  
a=\* (zero or more media attribute lines — overriding the Session attribute lines) optional fields are marked with an asterisk:

Alice’s scroll looked something like this: (Contents of Session Description Protocol (SDP))

v=0  
o=alice 2909090909 2909090909 IN IP4 host.anywhere.com  
s=  
c=IN IP4 host.anywhere.com  
t=0 0  
m=audio 49170 RTP/AVP 0  
a=rtpmap:0 PCMU/8000  
m=video 51372 RTP/AVP 31  
a=rtpmap:31 H261/90000  
m=video 53000 RTP/AVP 32  
a=rtpmap:32 MPV/90000

**Epilogue: The Digital Connection**

With the guidance of WebRTC, the understanding of the NAT Kingdom, the assistance of STUN and TURN, the wisdom of ICE, and the clarity of the SDP Scroll, Alice finally connected with Bob. Their browsers gleefully exchanged stories, smiles, and laughter, all thanks to the intricate dance of technologies behind the scenes.

And so, dear reader, the next time you video chat with someone across the digital expanse, remember the epic journey that your data undertakes to make that connection possible.