进击的 WebRTC

Web Real-Time Communications

FE-智慧商业

主题大纲

- 什么是 WebRTC ?
- WebRTC 有哪些应用场景?
- WebRTC 有哪些优缺点?
- WebRTC 核心组件和架构
- WebRTC 连接过程
- WebRTC 浏览器 API
- 交互式连接建立（ICE）& NAT 穿越

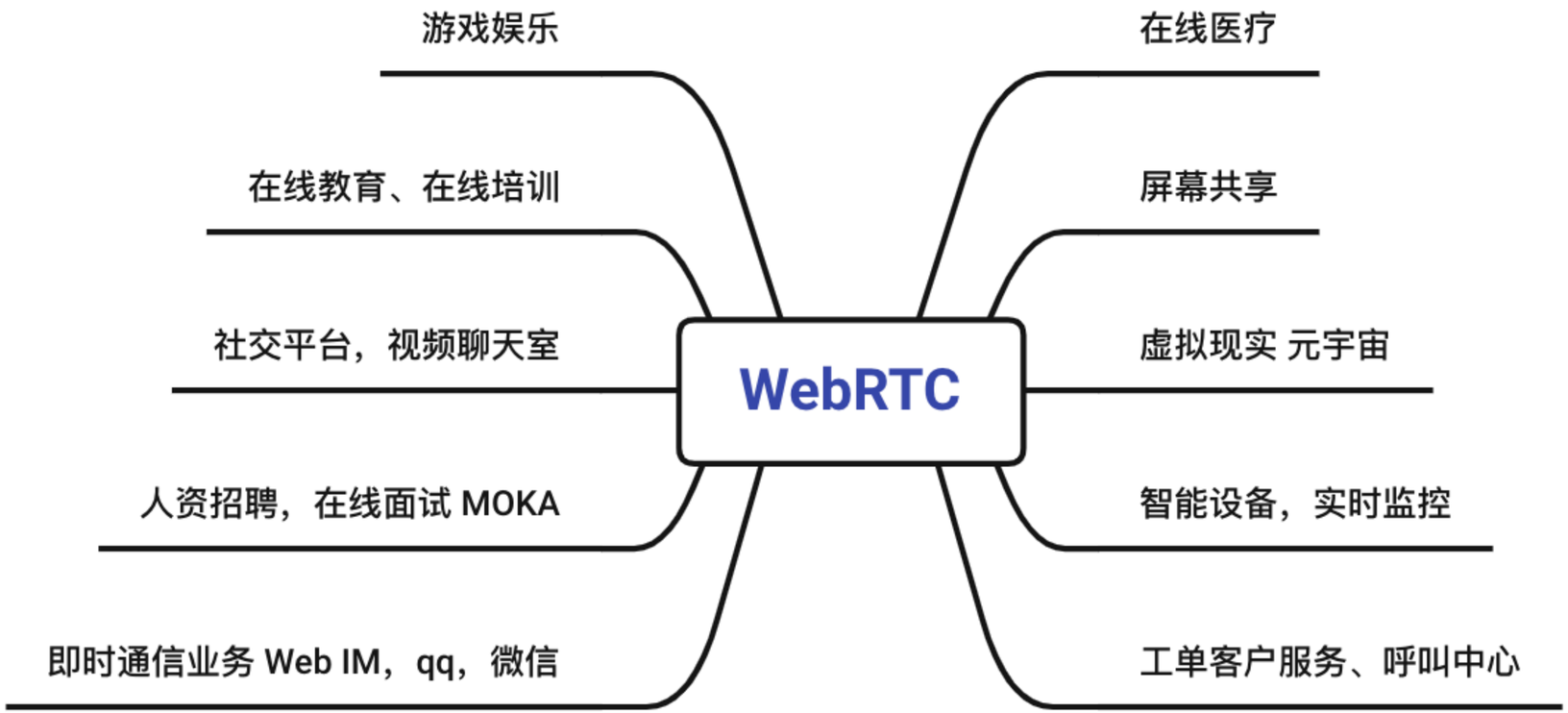
什么是 WebRTC?

WebRTC 是一个由 Google、Mozilla、Opera 等发起的开源项目，名称源自网页即时通信（Web Real-Time Communication）的缩写 。

其实 WebRTC 在不同场景下包含不同的含义，它既可以代表 Google 开源的 WebRTC 项目，又可以代表 W3C（World Wide Web Consortium-万维网联盟） 工作组制定的 WebRTC 标准，也可以代表浏览器中的 WebRTC 接口，我们将他们统称为 WebRTC 技术。

大多数时候，对于开发者而言 WebRTC 是一套支持网页浏览器进行实时音视频对话的 W3C Javascript API，它包括了音视频的采集、编解码、网络传输、显示等功能。

应用场景


游戏娱乐
在线教育、在线培训
社交平台，视频聊天室
人资招聘，在线面试 MOKA
即时通信业务 Web IM，qq，微信
在线医疗
屏幕共享
虚拟现实 元宇宙
智能设备，实时监控
工单客户服务、呼叫中心
WebRTC

优缺点

优点

1. **方便**。对于用户来说，在WebRTC出现之前想要进行实时通信就需要安装插件和客户端，这是一个复杂的过程。现在，WebRTC技术内置于浏览器中，用户不需要使用任何插件或者软件就能通过浏览器来实现实时通信。对于开发者来说，在Google将WebRTC开源之前，浏览器之间实现通信的技术是掌握在大企业手中，这项技术的开发是一个很困难的任务，现在开发者使用简单的HTML标签和JavaScript API就能够实现Web音/视频通信的功能。
2. **跨平台**。因为基于浏览器，所以可跨浏览器（浏览器支持 WebRTC）和跨操作系统平台，windows、Linux、ios、android……全部支持。
3. **P2P的优势**。使用P2P技术处理数据（音频、视频和文件等）的传输，可减少服务器端的性能压力和带宽成本。
4. **免费**。虽然WebRTC技术已经较为成熟，其集成了最佳的音/视频引擎，十分先进的codec，但是Google对于这些技术不收取任何费用。
5. **强大的打洞能力**。WebRTC技术包含了使用STUN、ICE、TURN、RTP-over-TCP的关键NAT和防火墙穿透技术，并支持代理。

缺点

1. WebRTC中很多的参数都是由GIPS公司的工程师们依靠经验所设定的值，这就会出现卡顿、延时、回声、丢包、多人视频不稳定等问题，并且由于公网的稳定性或机型适配等外在因素，以上问题在项目上线后会更加严重。

2. WebRTC缺乏服务器方案的设计和部署。

3. 传输质量难以保证。WebRTC的传输设计基于P2P，难以保障传输质量，优化手段也有限，只能做一些端到端的优化，难以应对复杂的互联网环境。比如对跨地区、跨运营商、低带宽、高丢包等场景下的传输质量基本是靠天吃饭，而这恰恰是国内互联网应用的典型场景。

4. WebRTC比较适合一对一的单聊，虽然功能上可以扩展实现群聊，但是没有针对群聊，特别是超大群聊进行任何优化。

5. 设备端适配，如回声、录音失败等问题层出不穷。这一点在安卓设备上尤为突出。由于安卓设备厂商众多，每个厂商都会在标准的安卓框架上进行定制化，导致很多可用性问题（访问麦克风失败）和质量问题（如回声、啸叫）。

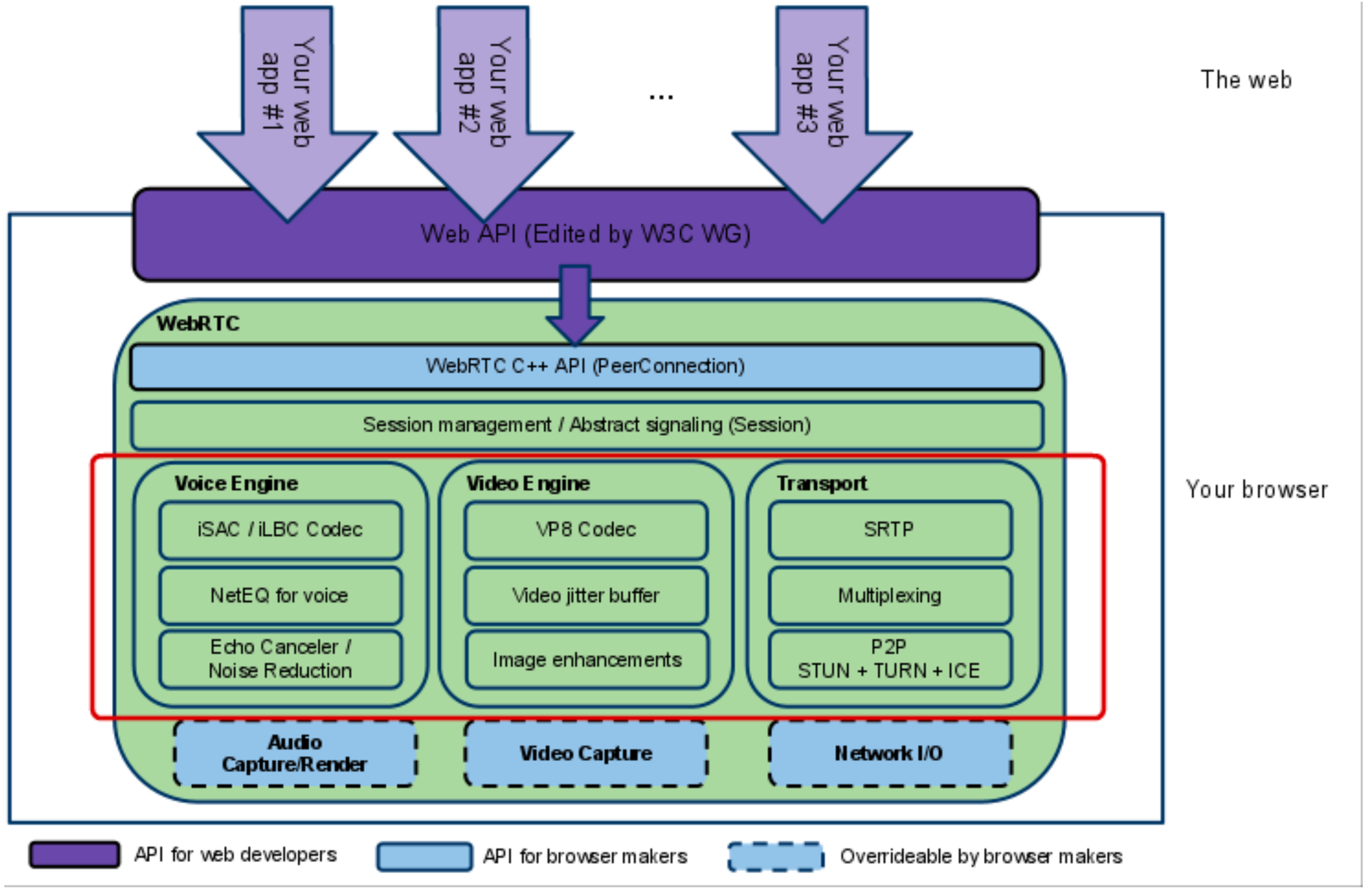
6. 对Native开发支持不够。WebRTC顾名思义，主要面向Web应用，虽然也可以用于Native开发，但是由于涉及到的领域知识（音视频采集、处理、编解码、实时传输等）较多，整个框架设计比较复杂，API粒度也比较细，导致连工程项目的编译都不是一件容易的事。

核心组件和架构

核心组件

1. 音视频引擎：OPUS、VP8 / VP9、H264
2. 传输层协议：底层传输协议为 UDP
3. 媒体协议：SRTP / SRTCP
4. 数据协议：DTLS / SCTP
5. P2P 内网穿透：STUN / TURN / ICE / Trickle ICE
6. 信令与 SDP 协商：HTTP / WebSocket / SIP、 Offer Answer 模型

架构

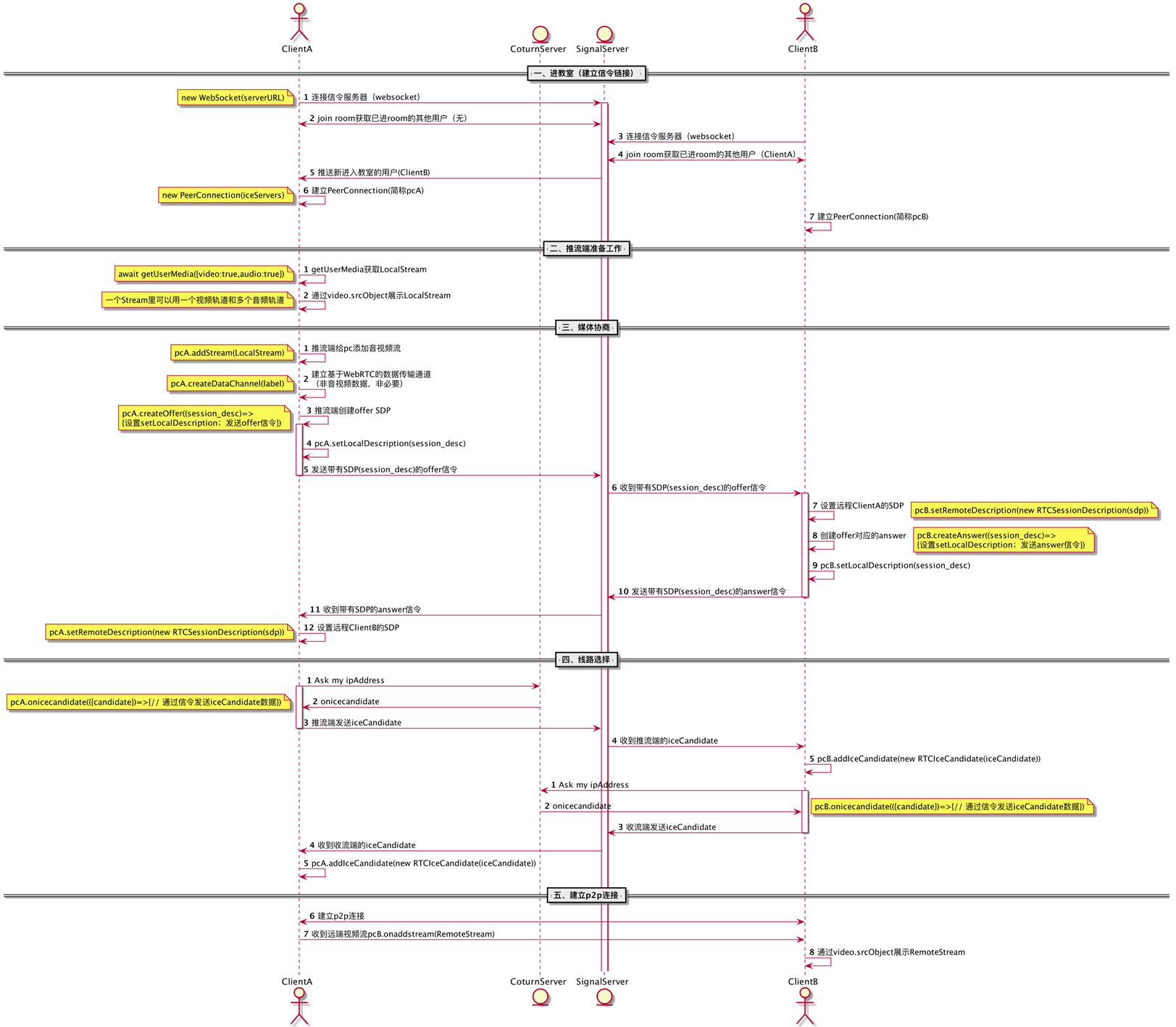

Your web app #1
Your web app #2
...
Your web app #3
The web
Web API (Edited by W3C WG)
WebRTC
WebRTC C++ API (PeerConnection)
Session management / Abstract signaling (Session)
Voice Engine
iSAC / iLBC Codec
NetEQ for voice
Echo Canceler / Noise Reduction
Video Engine
VP8 Codec
Video jitter buffer
Image enhancements
Transport
SRTP
Multiplexing
P2P STUN + TURN + ICE
Audio Capture/Render
Video Capture
Network I/O
Your browser
API for web developers
API for browser makers
Overrideable by browser makers

WebRTC 连接过程

WebRTC 连接过程

1. 从本地设备（如麦克风、网络摄像头）创建一个 MediaStream 对象。
2. 从本地 MediaStream 获取 URL Blob
3. 使用获取的 URL Blob 进行本地预览
4. 创建一个 RTCPeerConnection 对象
5. 将本地流添加到新创建的连接
6. 将你自己的会话描述发送到远程对等点 Send your own session description to the remote peer.
7. 从您的对等方接收远程会话描述 Receive the remote session description from your peer.
8. 处理收到的会话描述，并将远程流添加到您的 RTCPeerConnection
9. 从远程流获取 URL Blob
10. 使用获取的 URL Blob 播放远程对等方的音频和/或视频

WebRTC工作时序图



浏览器 API

浏览器 API

- MediaStream
- RTCPeerConnection
- RTCDataChannel

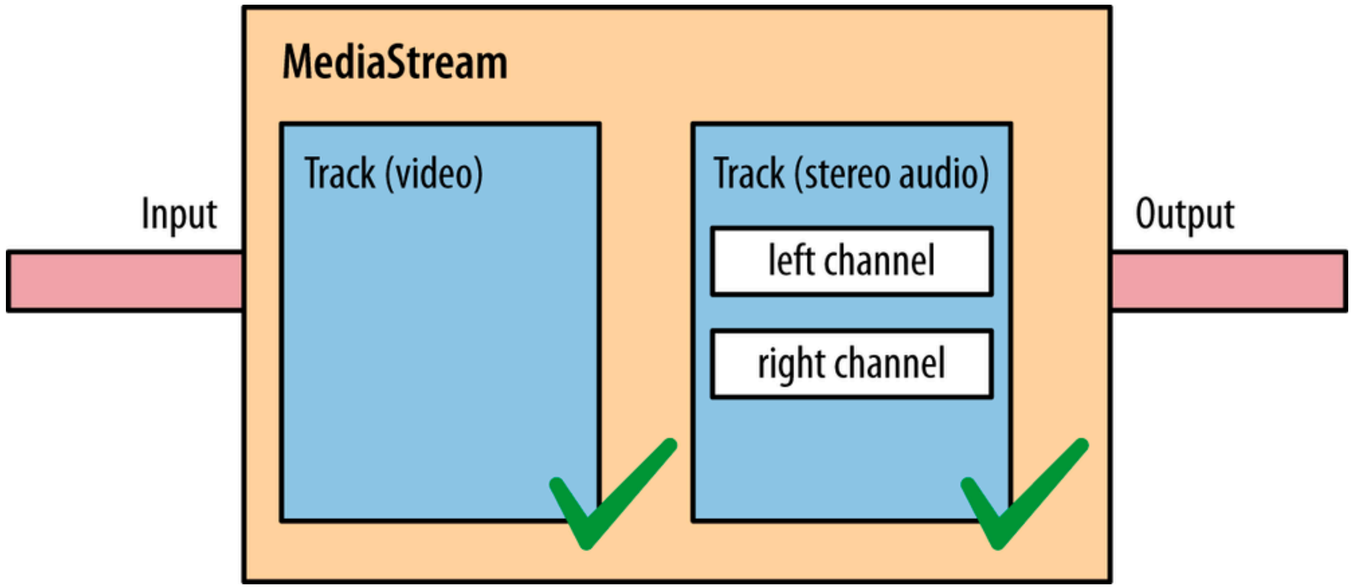
MediaStream

`MediaStream` 接口用于表示媒体数据流。 流可以是输入或输出，也可以是本地或远程（例如，本地网络摄像头或远程连接）。

必须注意，单个 `MediaStream` 可以包含零个或多个轨道。 每个轨道都有一个对应的 `MediaStreamTrack` 对象，该对象代表用户代理中的特定媒体源。

`MediaStream` 中的所有轨道在渲染时进行同步。

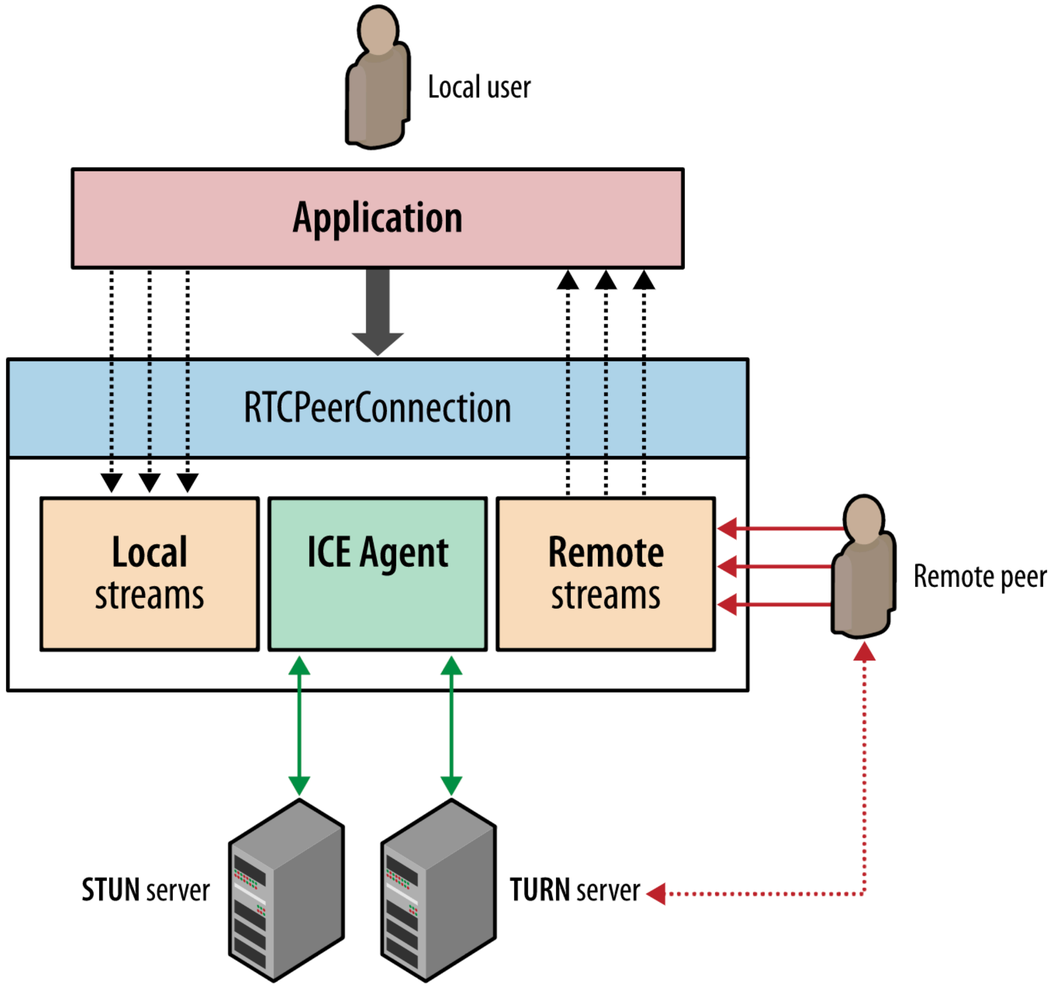
`MediaStreamTrack` 表示包含一个或多个通道的内容，其中，通道之间具有定义的已知的关系。 通道是此 API 规范中考虑的最小单位。 图右显示了由单个视频轨道和两个不同的音频（左声道和右声道）轨道组成的 `MediaStream`。



RTCPeerConnection

RTCPeerConnection 接口负责维护每一个端到端连接的完整生命周期

- 管理穿越 NAT 的完整 ICE工作流
- 发送自动（STUN）持久化信号
- 跟踪本地流
- 跟踪远程流
- 按需触发自动流协商
- 提供必要的 API，以生成连接提议，接收应答，允许我们查询连接的当前状态，等等



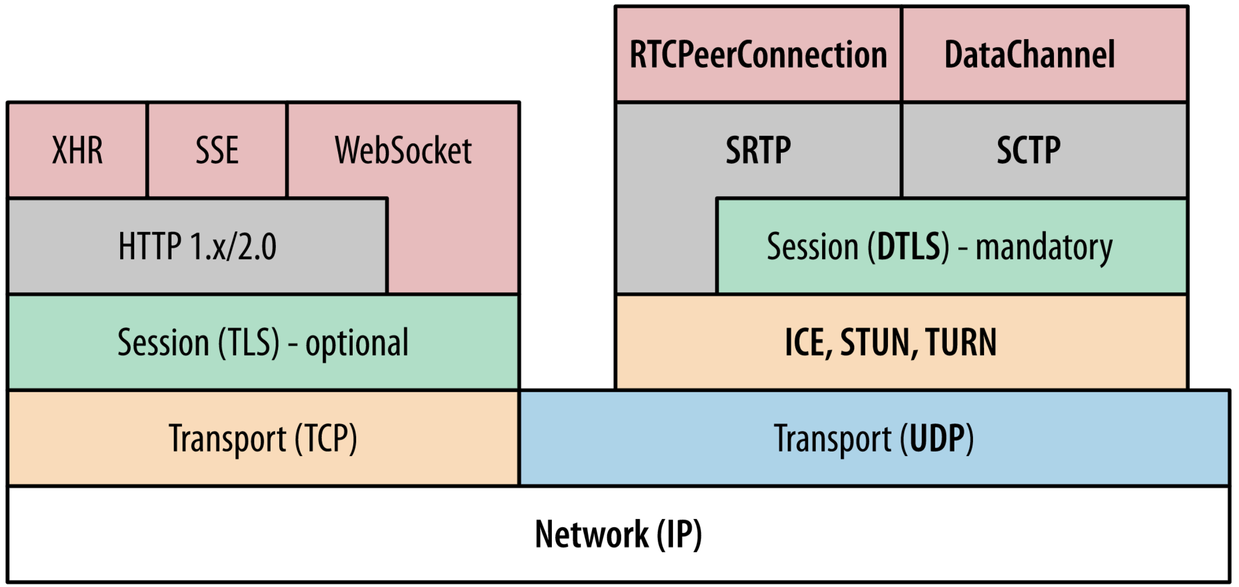
RTCDataChannel

建立在PeerConnection上的，不能单独使用

因为是浏览器间的直接通信，所以RTCDataChannel要比WebSocket快得多。

很多领域都潜在地使用到了这个API，比如：

1. 游戏
2. 远程桌面应用
3. 实时文字聊天
4. 文件传输
5. 分散网络



ICE & NAT 穿越

ICE Candidate

RFC5245 https://datatracker.ietf.org/doc/html/rfc5245

Host Candidate: A candidate obtained by binding to a specific port from an IP address on the host. This includes IP addresses on physical interfaces and logical ones, such as ones obtained through Virtual Private Networks (VPNs) and Realm Specific IP (RSIP) [RFC3102] (which lives at the operating system level).

这个地址来源于本地的物理网卡或逻辑网卡上的地址，对于具有公网地址或者同一内网的端可以用

Server Reflexive Candidate: A candidate whose IP address and port are a binding allocated by a NAT for an agent when it sent a packet through the NAT to a server. Server reflexive candidates can be learned by STUN servers using the Binding request, or TURN servers, which provides both a relayed and server reflexive candidate.

这个地址是端发送 Binding 请求到 STUN/TURN server 经过 NAT 时，NAT 上分配的地址和端口

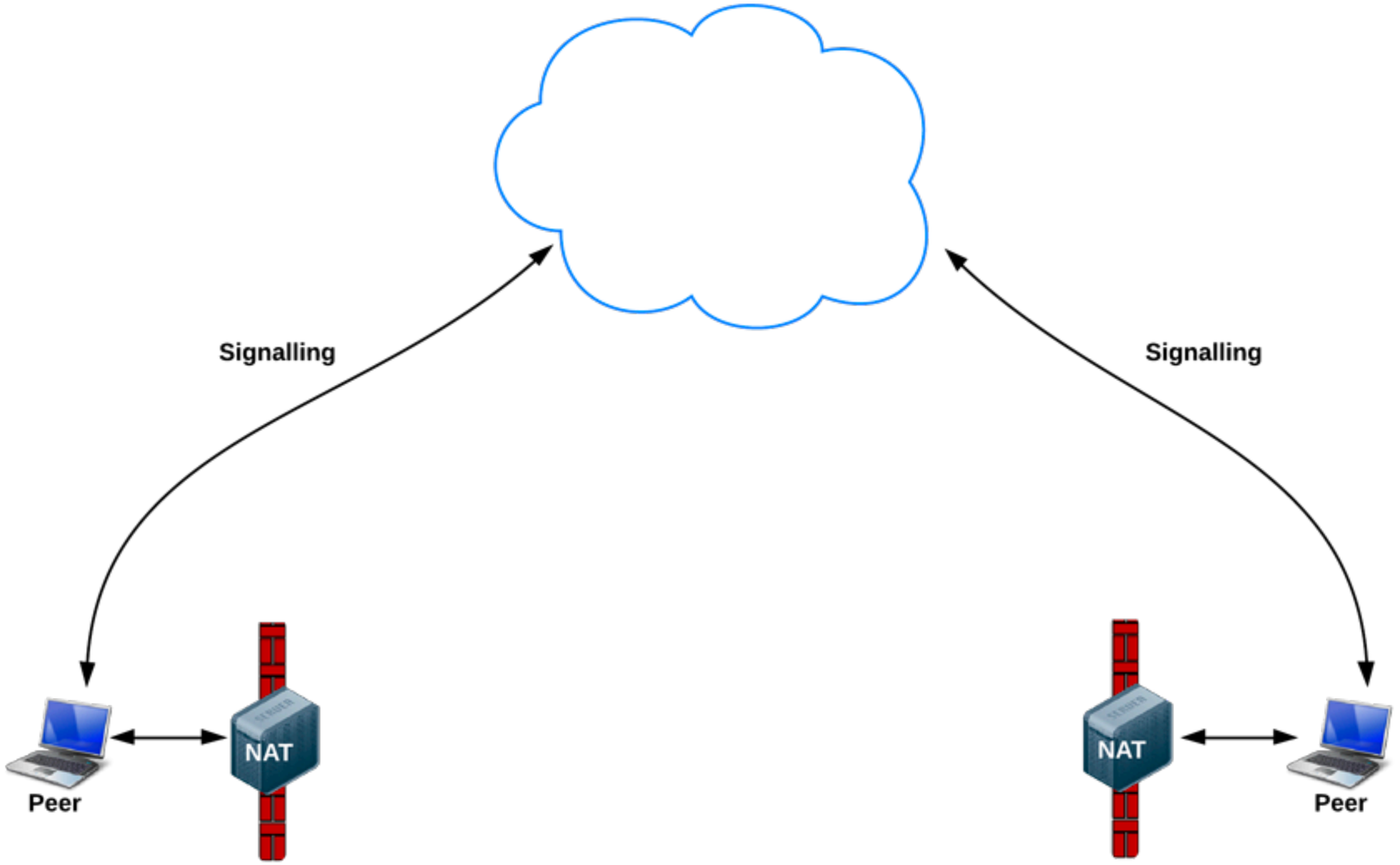
Peer Reflexive Candidate: A candidate whose IP address and port are a binding allocated by a NAT for an agent when it sent a STUN Binding request through the NAT to its peer.

这个地址是端发送 Binding 请求到对等端经过 NAT 时，NAT 上分配的地址和端口

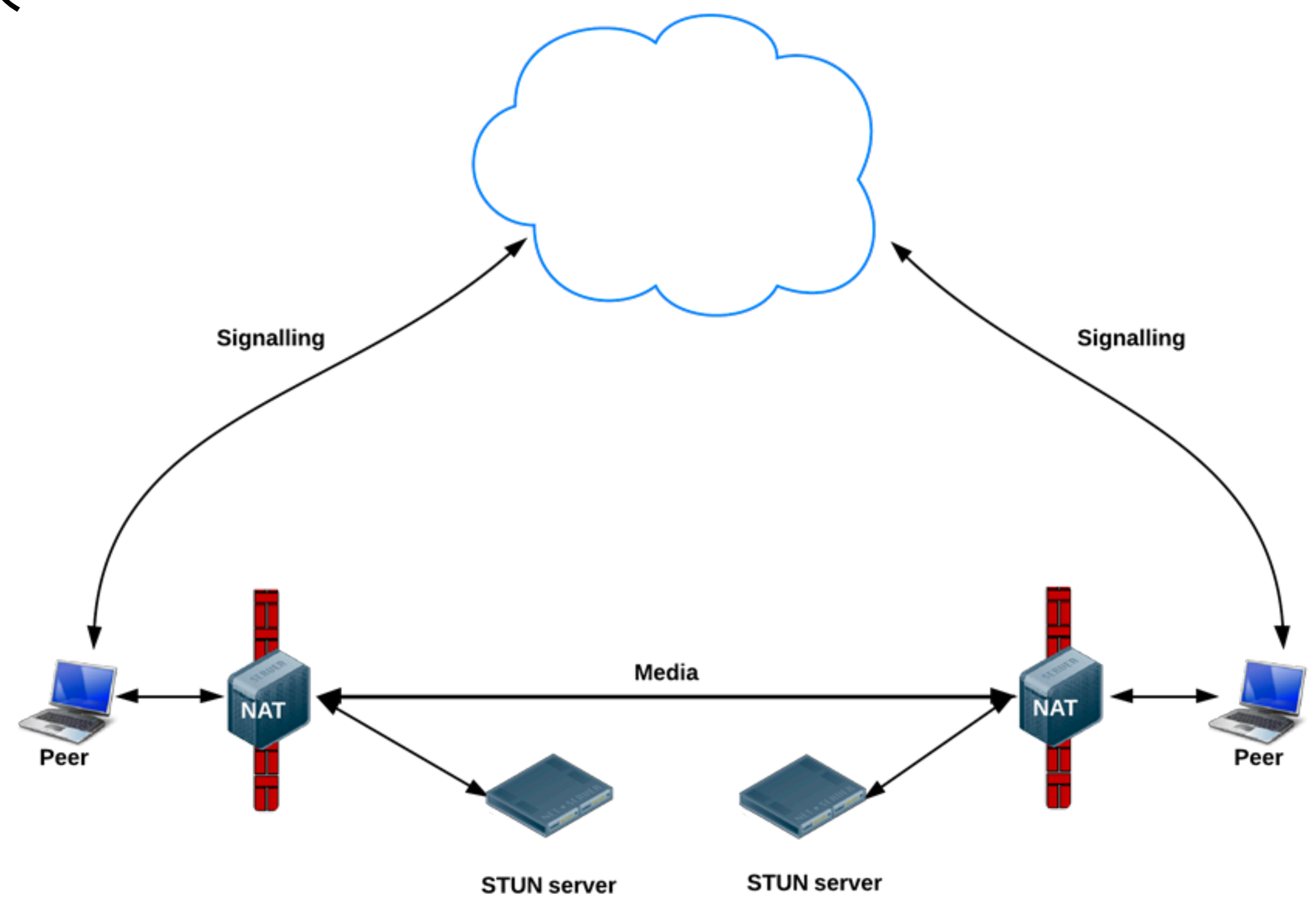
Relayed Candidate: A candidate obtained by sending a TURN Allocate request from a host candidate to a TURN server. The relayed candidate is resident on the TURN server, and the TURN server relays packets back towards the agent.

这个地址是端发送 Allocate 请求到 TURN server ，由 TURN server 用于中继的地址和端口

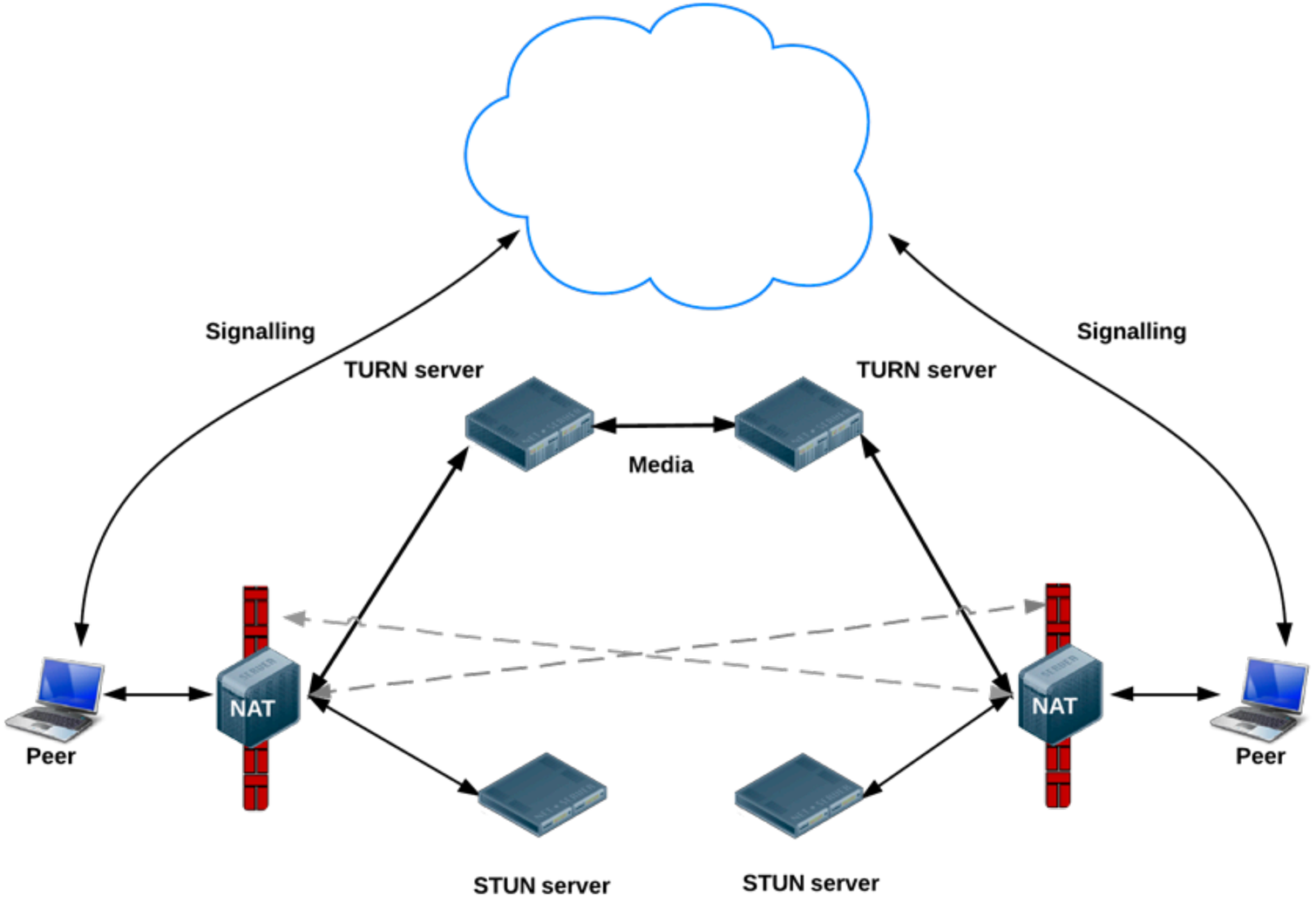
常见的网络拓扑

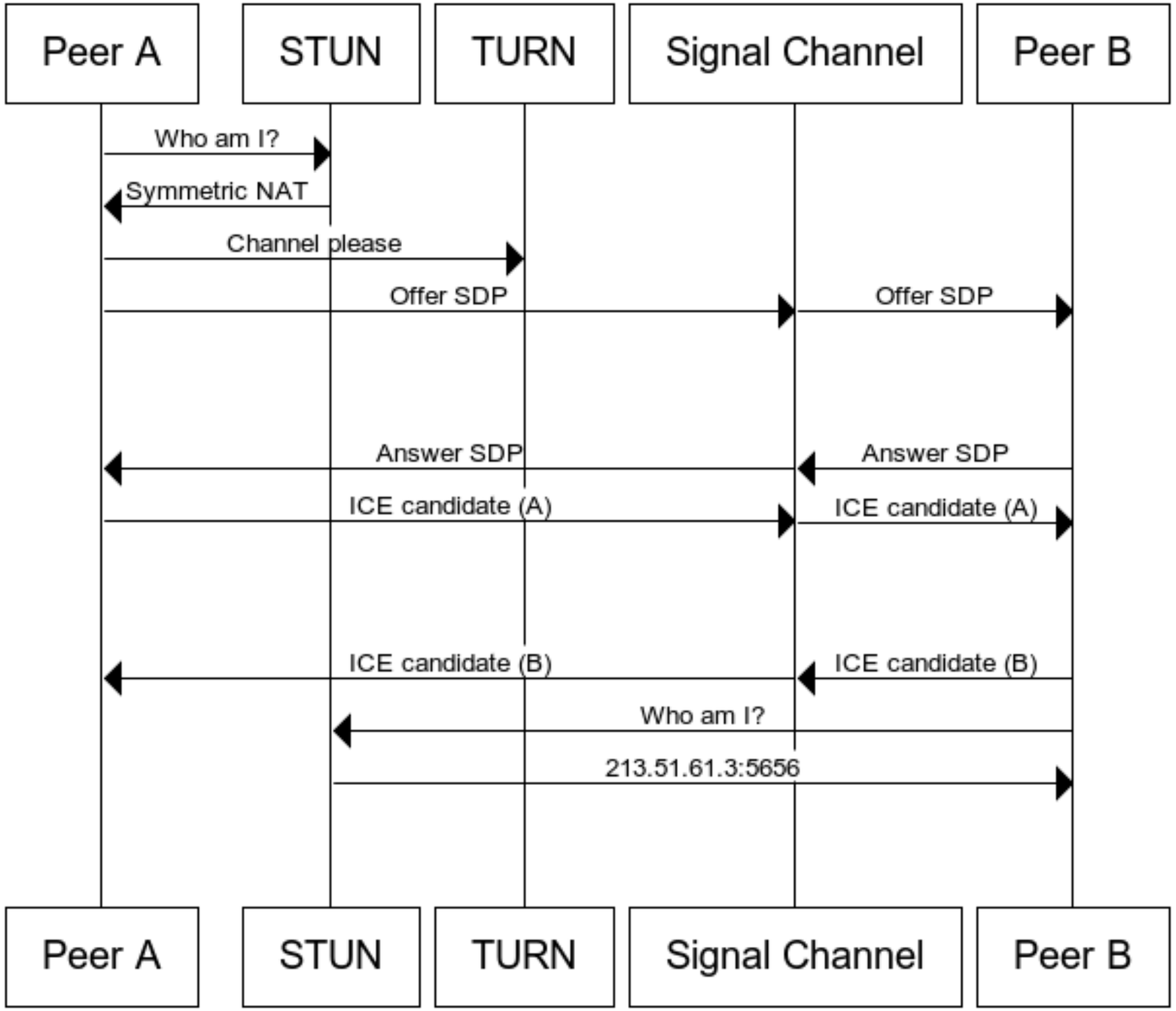


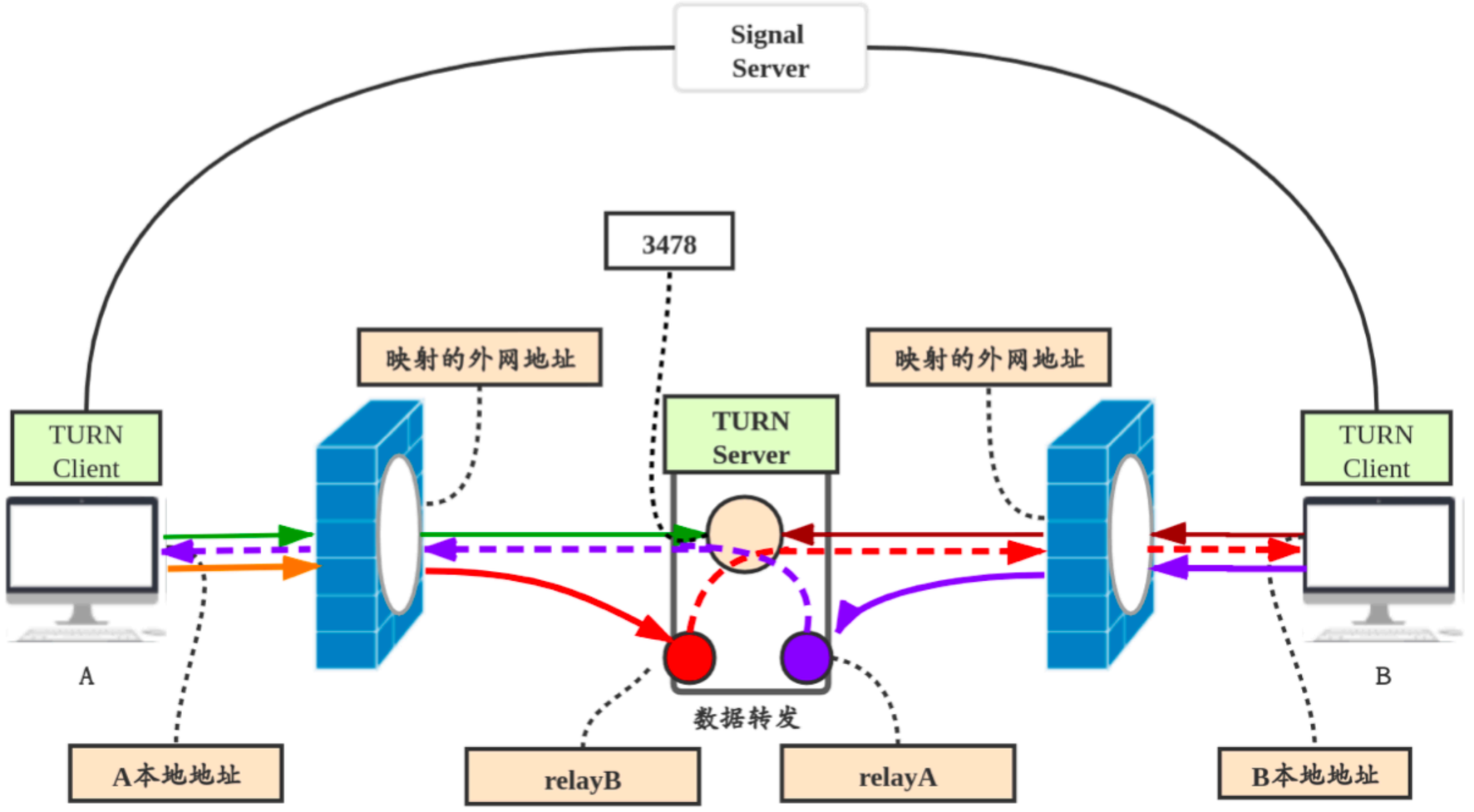
STUN 协议


Signalling
Signalling
Media
NAT
NAT
Peer
Peer
STUN server
STUN server

TURN 协议


Signalling
TURN server
TURN server
Signalling
Media
NAT
NAT
Peer
Peer
STUN server
STUN server


Peer A
STUN
TURN
Signal Channel
Peer B
Who am I?
Symmetric NAT
Channel please
Offer SDP
Offer SDP
Answer SDP
Answer SDP
ICE candidate (A)
ICE candidate (A)
ICE candidate (B)
ICE candidate (B)
Who am I?
213.51.61.3:5656
Peer A
STUN
TURN
Signal Channel
Peer B


Signal Server
3478
映射的外网地址
映射的外网地址
TURN Client
TURN Server
TURN Client
A
B
数据转发
A本地地址
relayB
relayA
B本地地址

NAT 穿越

RFC3489 : https://www.rfc-editor.org/rfc/rfc3489.txt
RFC5389 : https://www.rfc-editor.org/rfc/rfc5389.txt

1. 完全锥形 NAT
2. IP限制锥形 NAT
3. 端口限制 NAT
4. 对称型 NAT

完全锥型 NAT

完全锥型 NAT 的特点: 一旦打洞成功，所有知道该洞的主机都可以 通过它与内网主机进行通信。

{

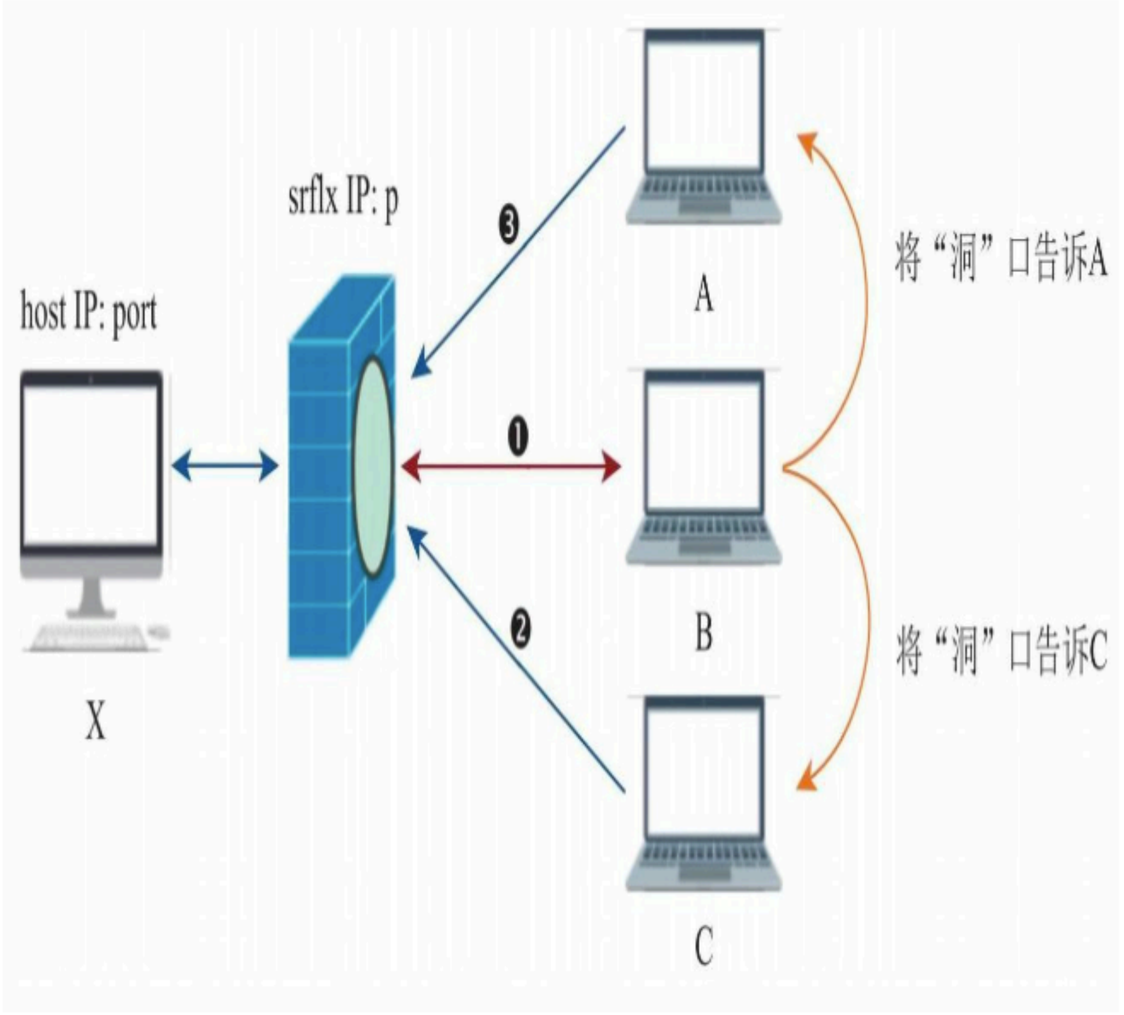
内网IP，

内网端口，

映射的外网IP，

映射的外网端口

}



IP 限制锥型 NAT

IP 限制锥型 NAT 的特点: 一旦打洞成功，只有与之打洞成功的外网主机才能通过该洞 与内网主机通信，
而其他外网主机即使知道洞口也不能与之通信。

{

内网IP，

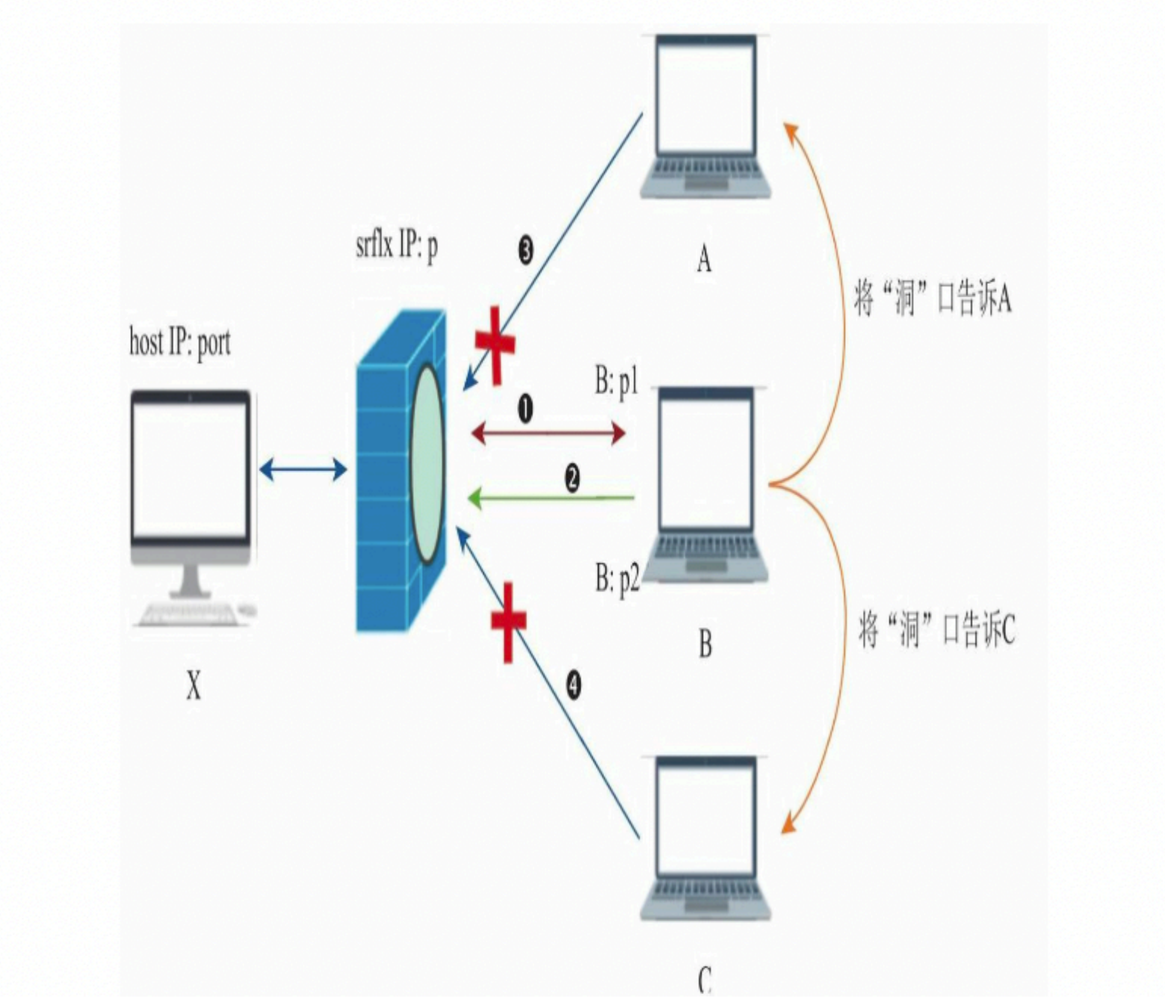
内网端口，

映射的外网IP，

映射的外网端口，

[被访问主机的IP，…]

}



端口限制锥型 NAT

端口限制锥型 NAT 的特点: 一旦打洞成功，除了像 IP 限制锥型 NAT 一样需要对IP地址进行检测外，

还需要对端口进行检测 。

{

内网IP ，

内网端口，

映射的外网IP ，

映射的外网端口，

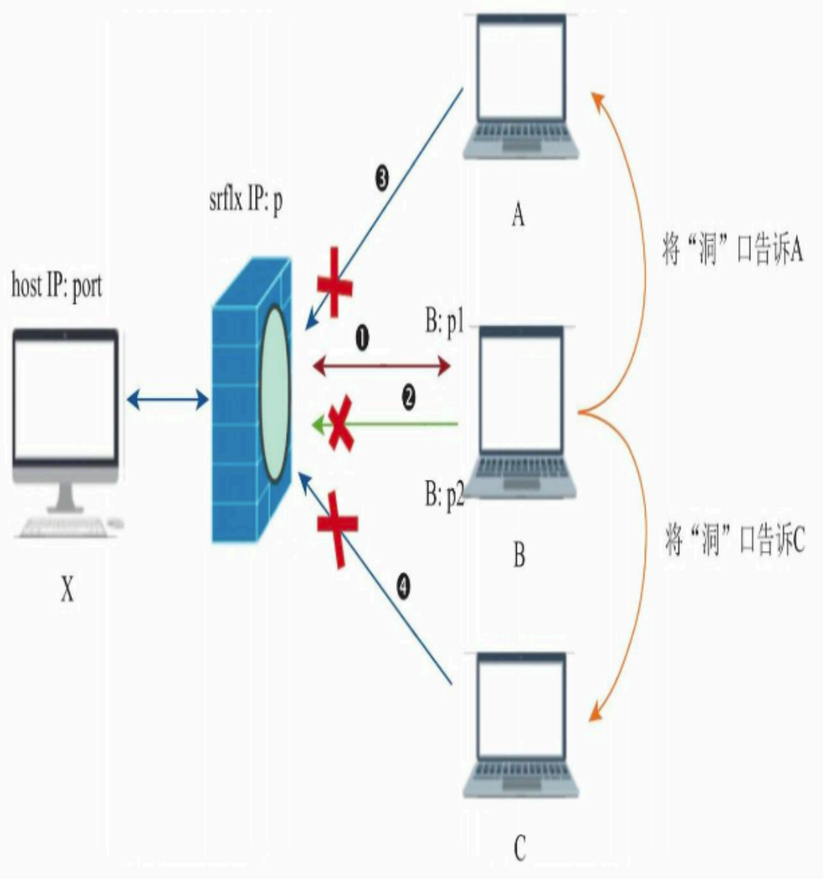
[

{被访问主机的IP，被访问主机的端口 }，

…

]

}



对称型 NAT

对称型 NAT 的特点: 内网主机每次访问不同的外网主机时，

都会生成一个新洞，而 不像前面3种NAT类型使用的是同一个洞。

{

内网IP ，

内网端口，

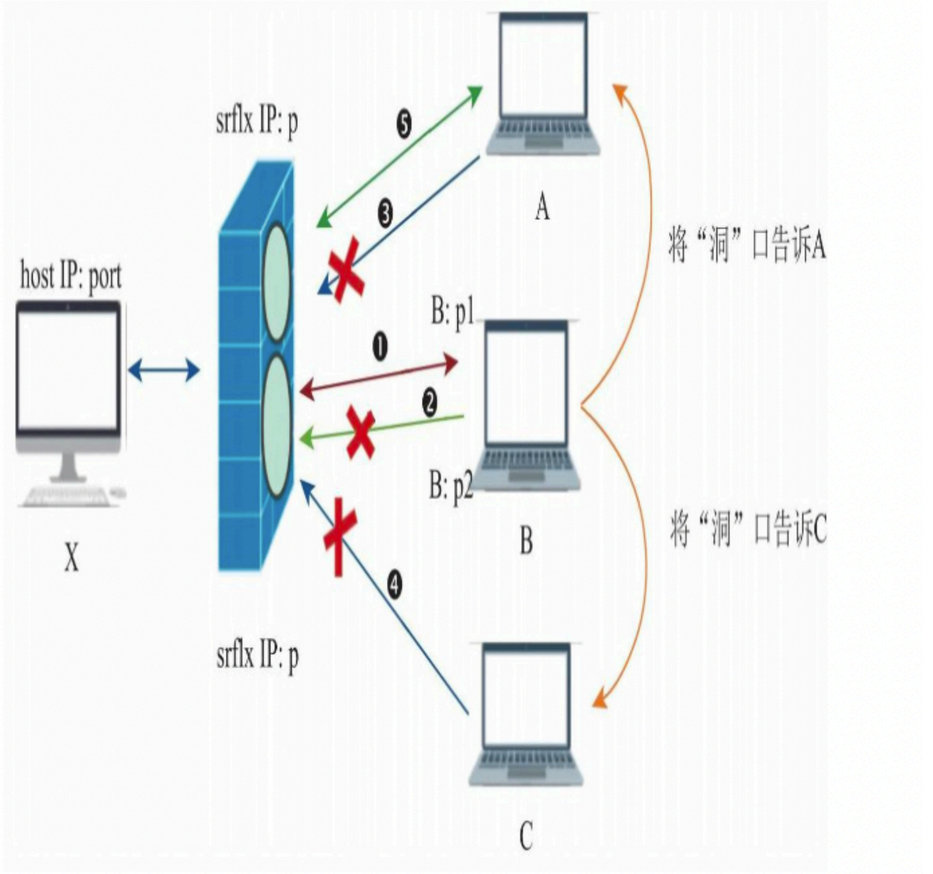
映射的外网IP ，

映射的外网端口，

被访问主机的IP，

被访问主机的端口

}



各 NAT 之间可穿越表

NAT 类型	NAT 类型	能否穿越
完全锥型 NAT	完全锥型 NAT	可以
完全锥型 NAT	IP 限制锥型 NAT	可以
完全锥型 NAT	端口限制锥型 NAT	可以
完全锥型 NAT	对称型 NAT	可以
IP 限制锥型 NAT	IP 限制锥型 NAT	可以
IP 限制锥型 NAT	端口限制锥型 NAT	可以
IP 限制锥型 NAT	对称型 NAT	可以
端口限制锥型 NAT	端口限制锥型 NAT	可以
端口限制锥型 NAT	对称型 NAT	不可以
对称型 NAT	对称型 NAT	不可以

仓库地址

Fronted：https://registry.code.tuya-inc.top/yexm/ty-chat-fronted

Backend：https://registry.code.tuya-inc.top/yexm/ty-chat-backend

参考文档

1. https://www.w3.org/TR/webrtc/#intro
2. https://www.w3.org/TR/webrtc-priority/#intro
3. https://bloggeek.me/psa-mdns-and-local-ice-candidates-are-coming/
4. https://webrtc.org/
5. https://www.infoq.cn/article/why-do-we-need-webrtc
6. https://hpbn.co/webrtc/

谢谢大家