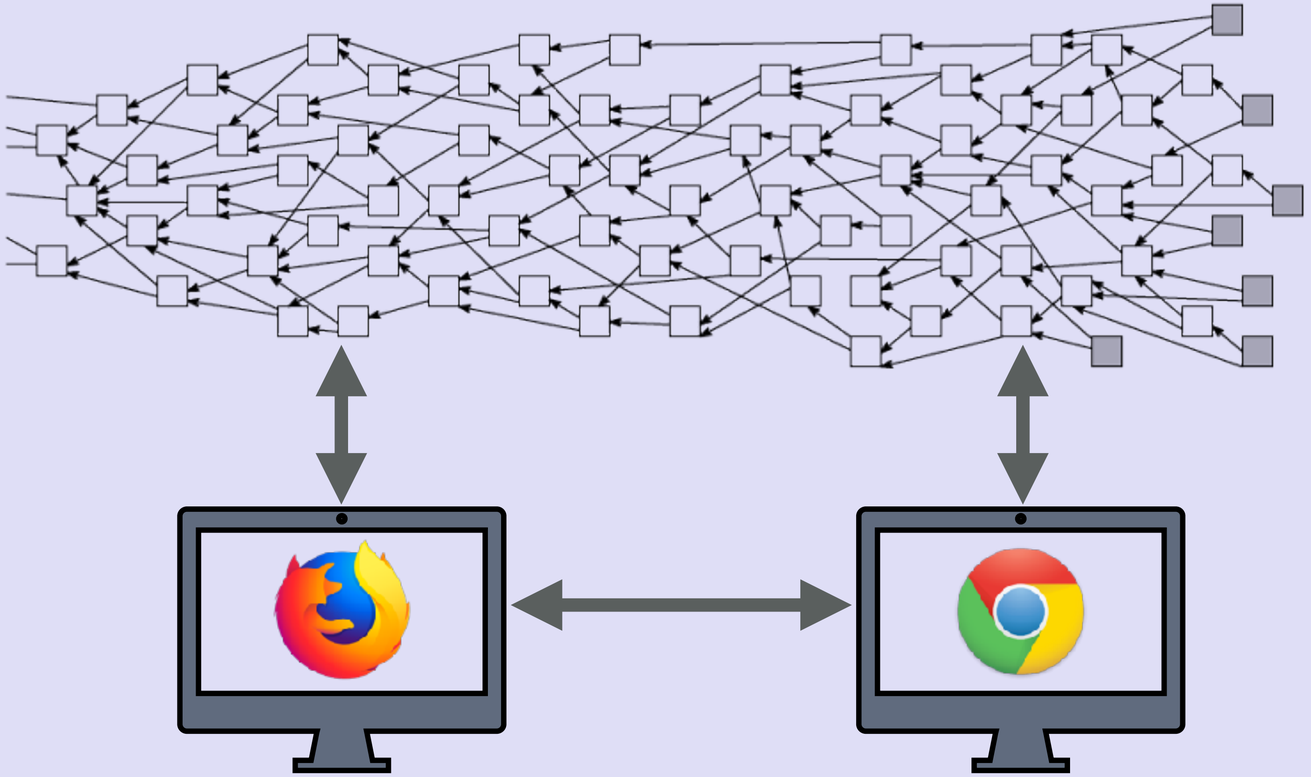
IOTA TUTORIAL 25

WebRTC & MAM Signaling



INTRO

- In this video I will give you a simplified explanation what WebRTC is and how it is used.

- I have created a proof-of-concept using MAM as a signaling implementation for WebRTC.

WEBRTC

- WebRTC (Web Real-Time Communication) was announced in 2011 and is a HTML5 specification supported by Google, Mozilla and Opera, amongst others.

- WebRTC provides browsers and devices with direct data, voice and video peer-to-peer communication without the need to install plugins or download native apps.

- WebRTC is supported by most modern browsers such as Chrome, Firefox, Safari and Microsoft Edge.

WEBRTC API

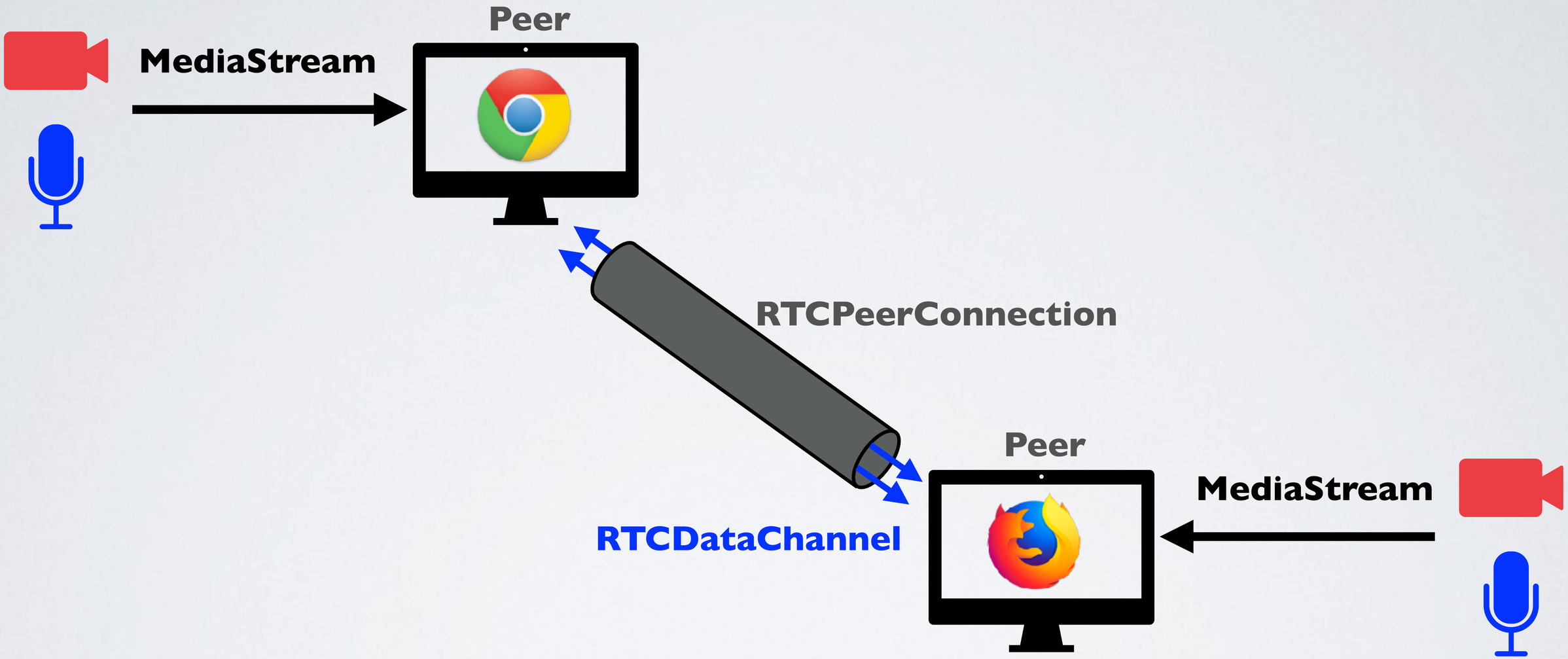
- WebRTC uses the following main component JavaScript APIs:

  - RTCPeerConnection
    To setup and create a peer-to-peer connection.

  - RTCDataChannel
    To bidirectional transfer arbitrary data peer-to-peer.
    Every data channel is associated with an RTCPeerConnection, and each peer connection can have one or more data channels.

  - MediaStream (more commonly known by its JavaScript function getUserMedia)
    It gives access to a stream object that represent video (camera) and audio (microphone) streams.

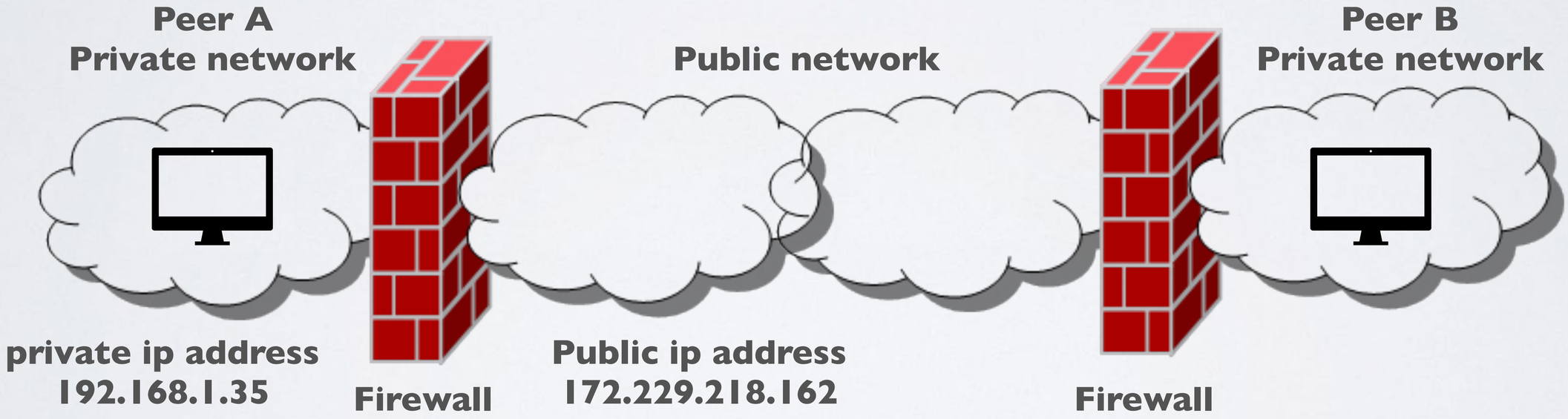
WEBRTC


Peer
MediaStream
RTCPeerConnection
Peer
MediaStream
RTCDataChannel

ICE, STUN AND TURN

- If two peers needs to communicate directly with each other they need to know each other public ip address and port.

- Often a direct connection is not possible because the peers uses a router with a built-in firewall that uses Network Address Translation (NAT).



ICE, STUN AND TURN

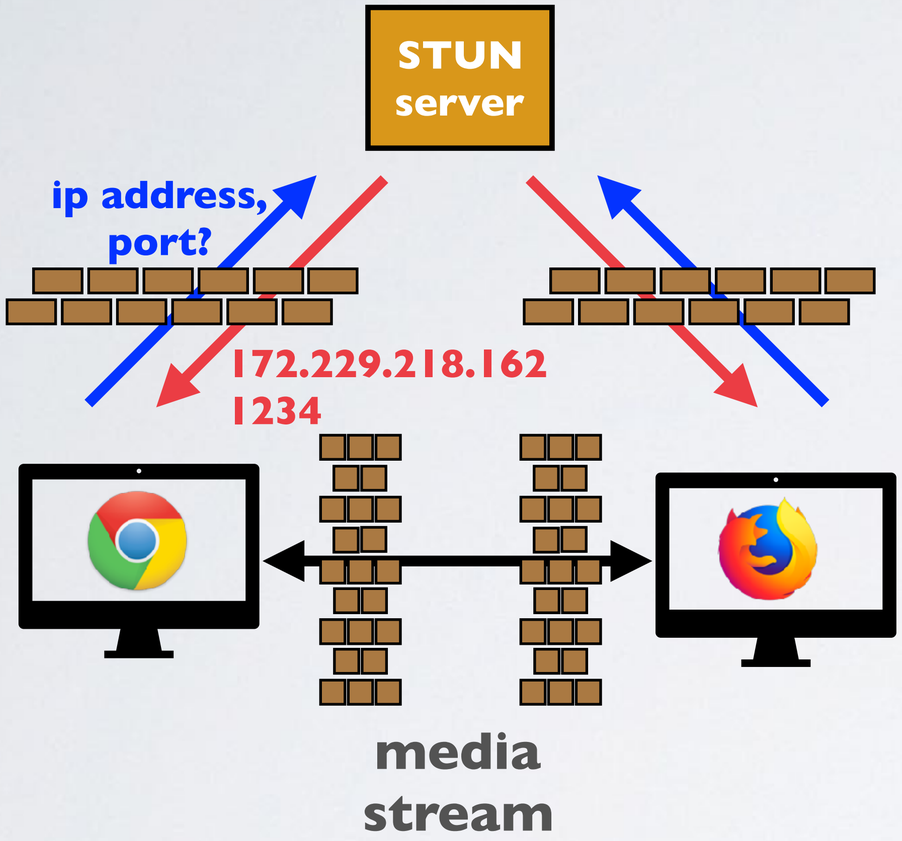
- The Interactive Connectivity Establishment framework (ICE) deals with the process of connecting peers through NATs.

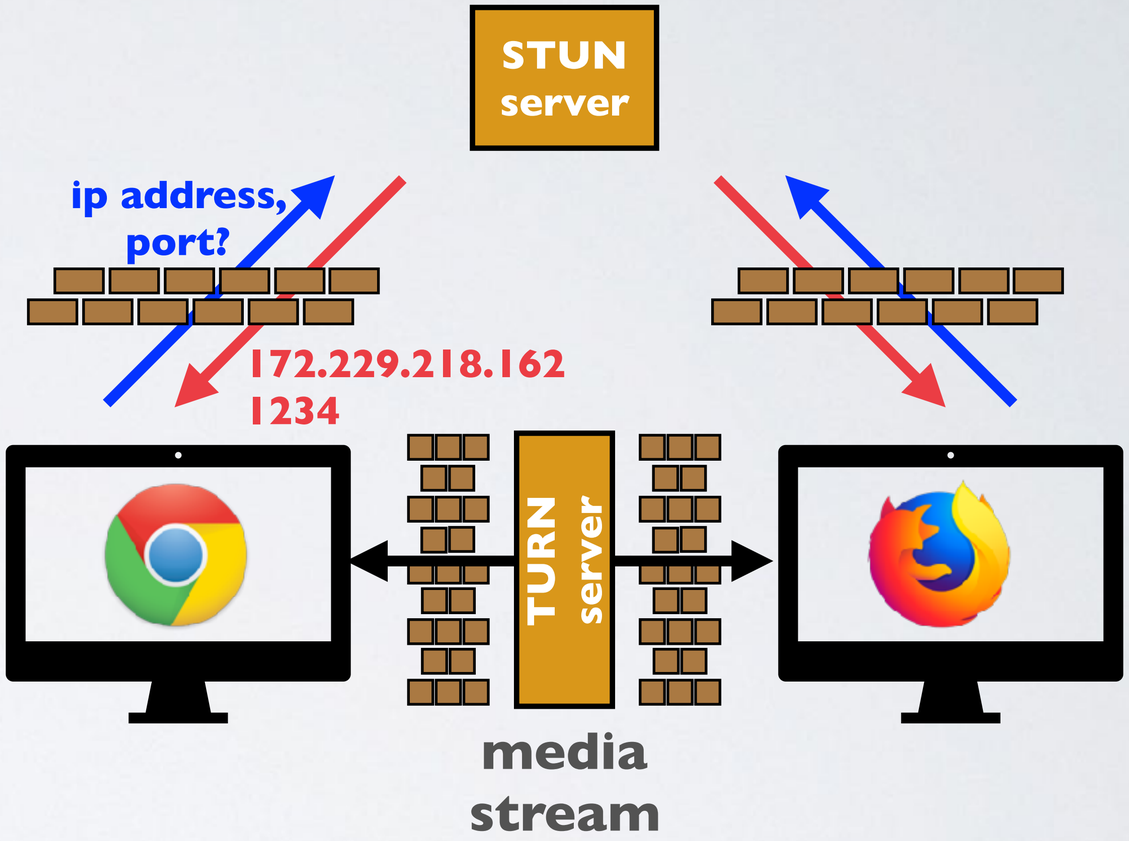
- A STUN (Session Transversal Utilities for NAT) server allows the peers to discover their public IP address, port and the type of NAT they are behind. This information is used to establish a peer-to-peer connection. A media stream will flow directly between the peers.

- In most cases (~70%) a STUN server suffice to setup a peer-to-peer connection. If a STUN server cannot establish the connection, ICE uses a TURN (Traversal Using Relay NAT) server.

- When a TURN server is used, this server relays the media stream between the peers.

ICE, STUN AND TURN





ICE, STUN AND TURN

- The use of a STUN server is preferred above a TURN server because a TURN server uses a lot of processing power.
- In a WebRTC application the STUN and TURN server locations can be specified.
- There are public STUN servers available but use them for prototyping or non-mission critical applications.

WEBRTC SIGNALING

- To create a peer-to-peer connection, the peers must also exchange several types of information first, for example:
  - Their external IP addresses and ports.
  - Their codecs and media types that they support.
  - When to initialise, close, and modify the communications sessions.
- This exchange of information between peers is called signaling and usually an external server is used called a "signaling server" which can store this information, for example in a database.
- Signaling methods and protocols are not specified by the WebRTC standards.

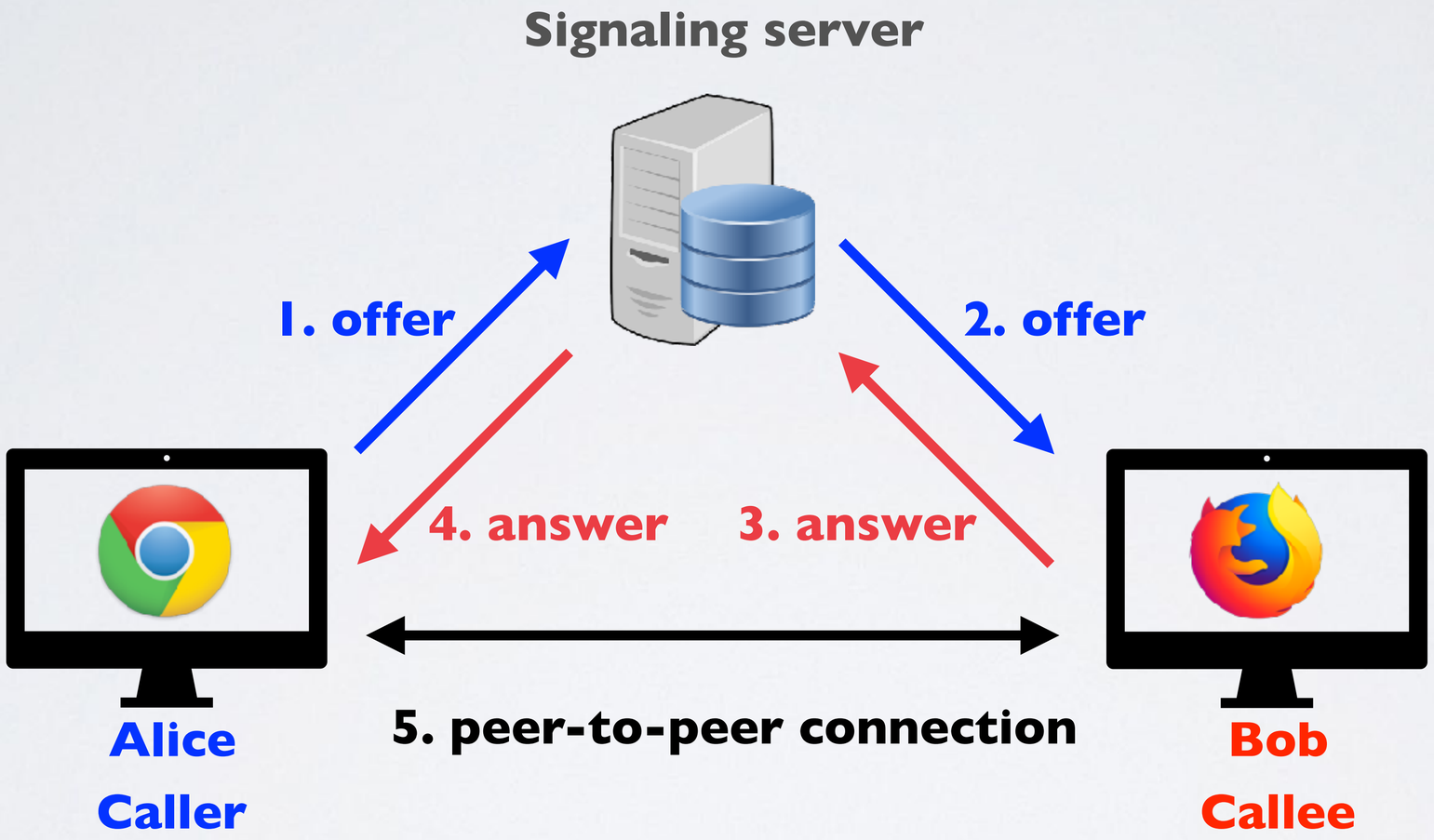
WEBRTC SIGNALING

- When Alice initiates a peer-to-peer communication with Bob, Alice is called the local user (aka caller) and Bob is the called the remote user (aka callee).

- The information send from Alice's browser to a signaling server is called the "offer", and Bob's browser information send to a signaling server is called the "answer".

- The offer and answer are written in a so called Session Description Protocol (SDP) format.

WEBRTC SIGNALING PROCEDURE



WEBRTC NO SIGNALING SERVER DEMO

- To demonstrate the WebRTC signaling process use the following application:
  https://www.mobilefish.com/download/webrtc/webrtc_noserver.html

MORE INFORMATION ABOUT WEBRTC

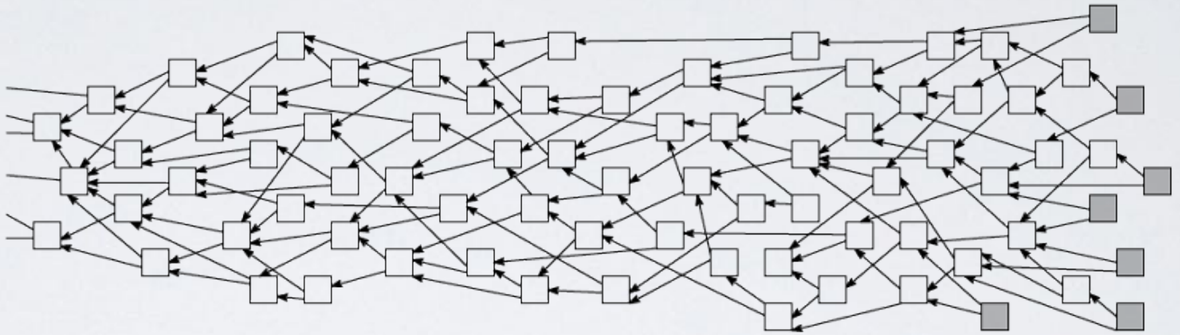
- https://webrtc.org
- https://www.html5rocks.com/en/tutorials/webrtc/basics/
- https://github.com/webrtc/samples
- https://www.tutorialspoint.com/webrtc/index.htm
- https://codelabs.developers.google.com/codelabs/webrtc-web/index.html
- https://www.w3.org/TR/webrtc/

MAM SIGNALING IMPLEMENTATION

- I have created a proof-of-concept to test if Masked Authenticated Messaging (MAM) can be used as a signaling implementation for WebRTC.

- See: https://www.mobilefish.com/services/cryptocurrency/mam_webrtc.html

MAM SIGNALING PROCEDURE



Signaling

3. publish seed:offer

2. Generate new seed

8. publish answer using the root

5. retrieve seed:offer

4. Calculate root and check for answer at regular time intervals

1. Create offer

9. retrieve answer using the root

6. Create answer

7. Calculate root

10. peer-to-peer connection

Alice

Caller

Bob

Callee

LESSONS LEARNED

- You can use Masked Authenticated Messaging (MAM) as a signaling implementation for WebRTC. However it takes too long to establish a peer-to-peer connection because publishing the offer and answer to the Tangle takes too much time.

- In a production like environment this is not acceptable, but for prototyping or just for demo applications its perfect.

- When using MAM it is recommended to compress your data, which will decrease the time to publish this data to the Tangle.
  For example you can use the lz-string compression javascript library.
  See: http://pieroxy.net/blog/pages/lz-string/index.html