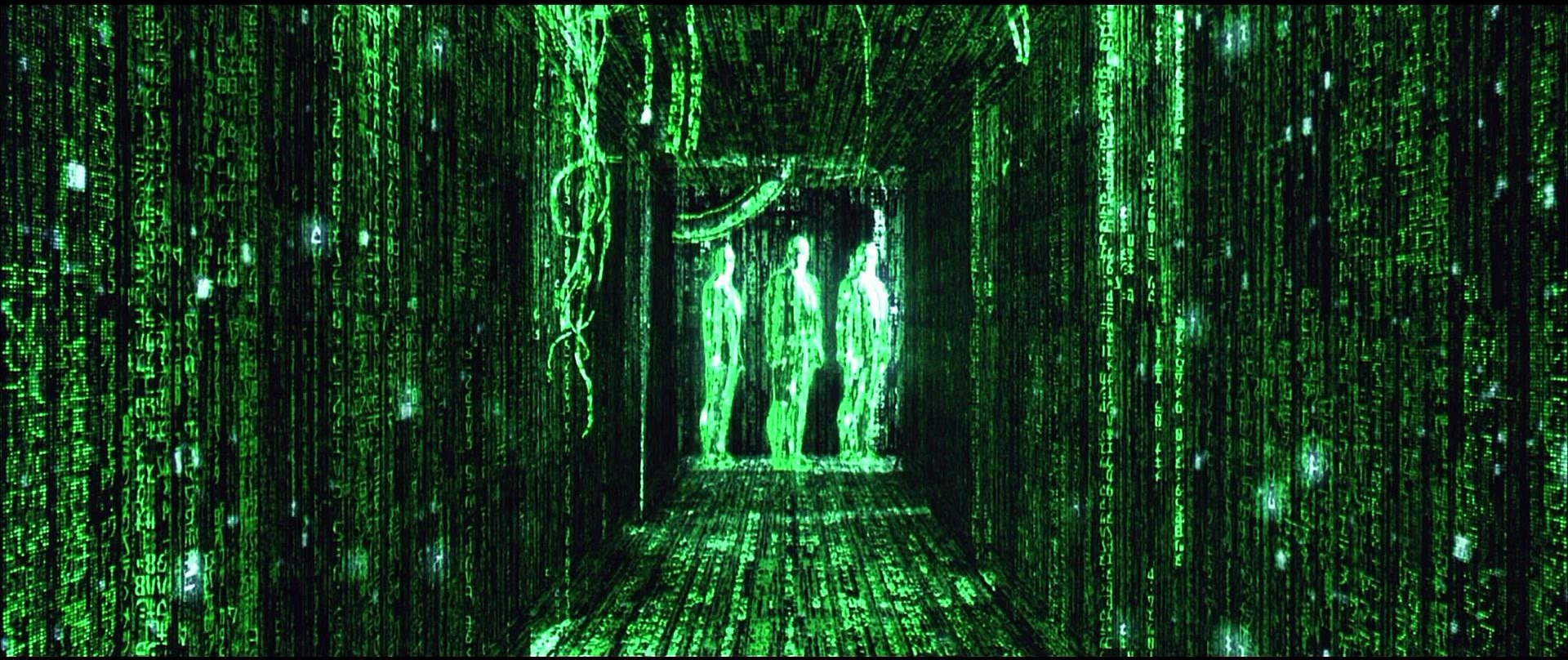
WebRTC

Web Real Time Communication

by Juan Roldán



WebRTC on Chrome

- Bringing real-time communications to the web
- Building a state-of-the-art media stack into Chrome
- Developing a new communications platform

Benefits

- This technology is **cross-browser** and is available for being used in all browsers.
- There is **no need** to install any plugin inside the browser to use it.
- There are a lot of **examples** around that can be used to implement a simple video/chat application.

APIs inside the browser

JavaScript APIs

- MediaStream (aka getUserMedia)
- RTCPeerConnection
- RTCDataChannel

MediaStream (aka getUserMedia)

getUserMedia

Basic JS code required to start

http://www.simpl.info/getusermedia/ (main.js)

Packages sent and received

http://www.simpl.info/rtcdatachannel/

RTCPeerConnection

Audio and video communication between peers

RTCPeerConnection

- Signal processing
- Codec handling
- Peer to peer communication
- Security
- Bandwidth management

RTCDataChannel

Bidirectional communication of arbitrary data between peers

RTCDataChannel

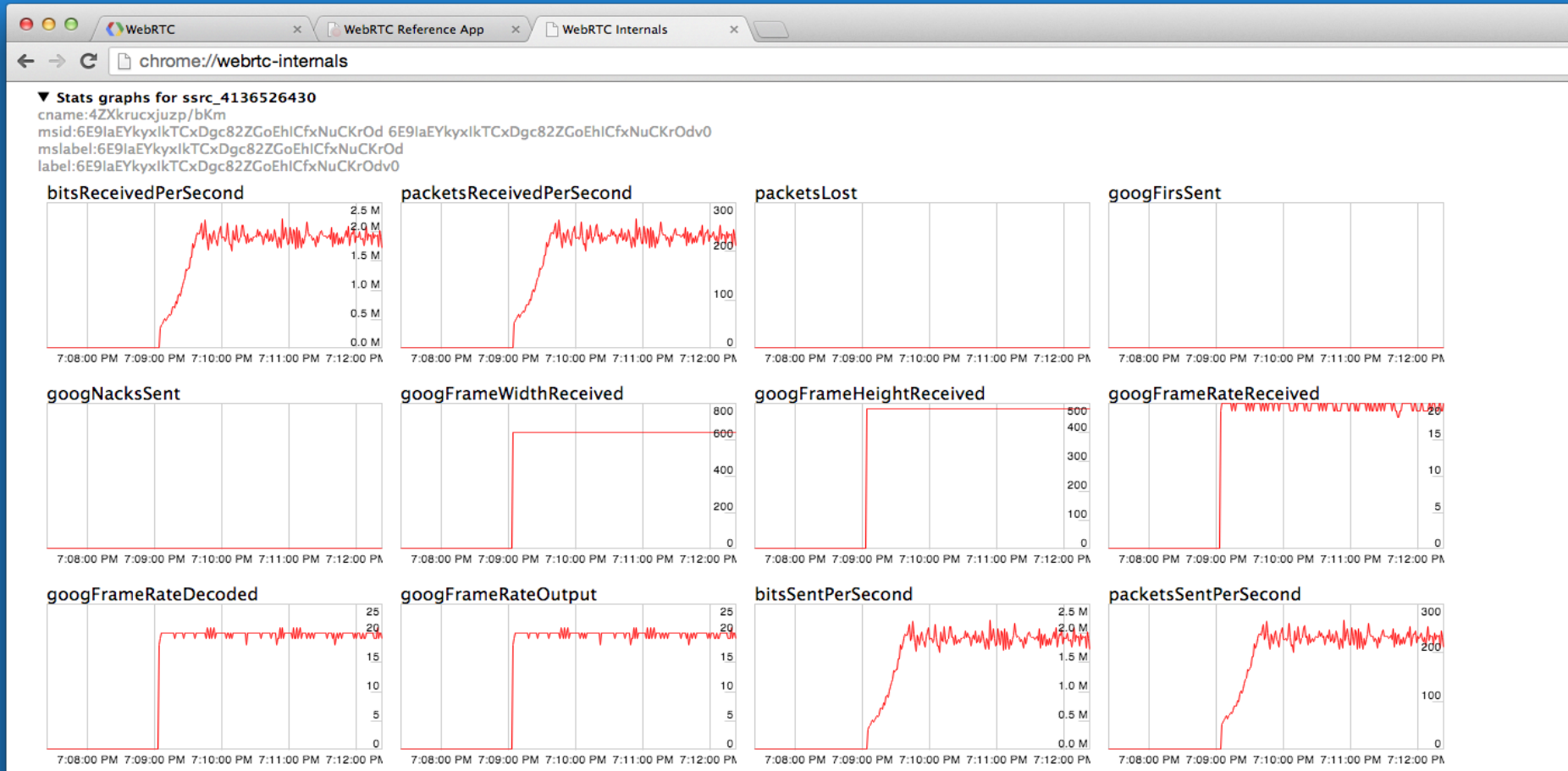
- Same API as WebSockets
- Ultra-low latency
- Unreliable or reliable
- Secure

“WebRTC is a new front in the long war for an open and unencumbered web”

Brendan Eich

– Mozilla CTO and inventor of JavaScript

chrome://webrtc-internals



Servers

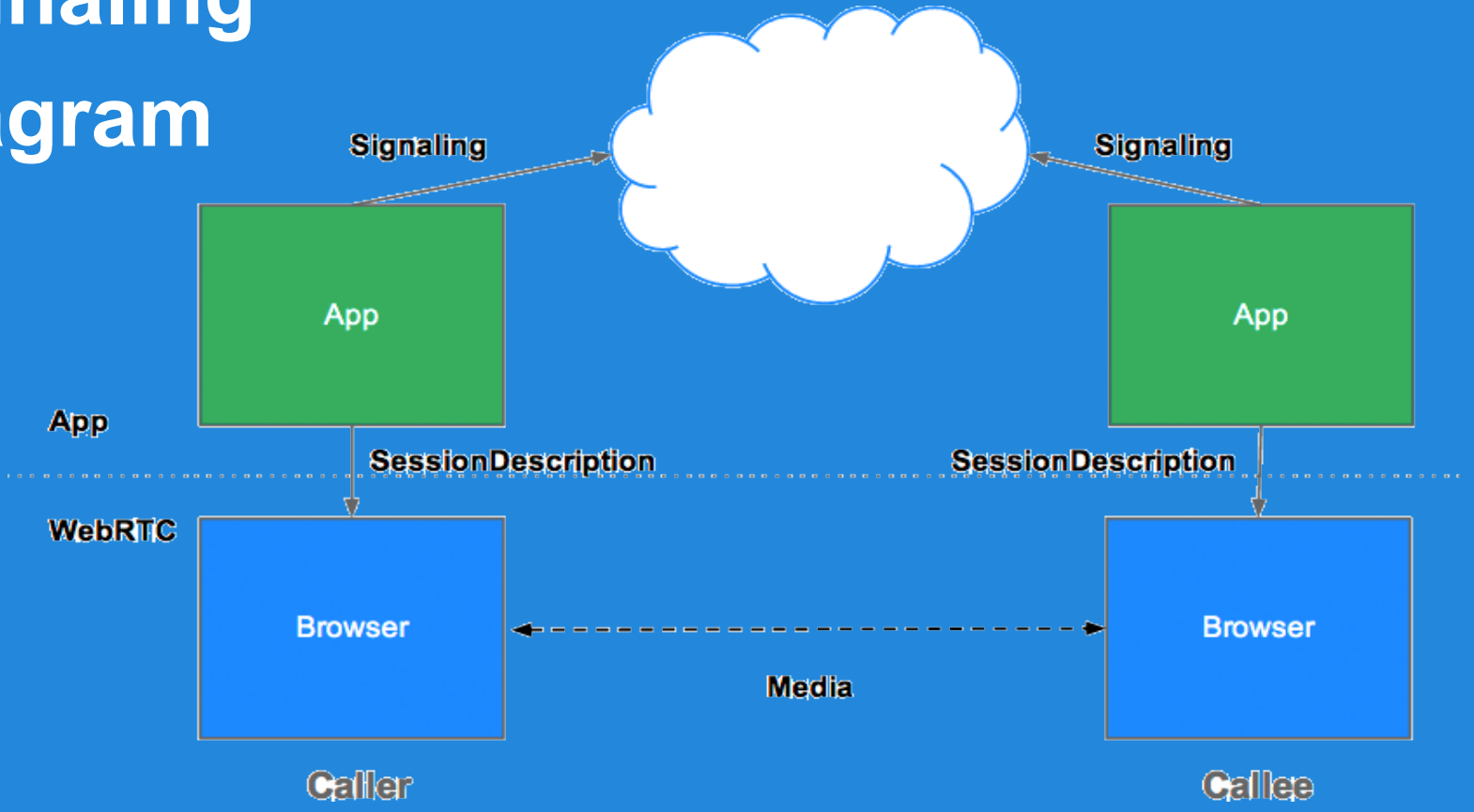
“stun” - “turn” - Diagrams

Servers

Once WebRTC implemented the last step inside the setup is deploying and configuring "servers" to handle the communication between the users involved.

- Handle the synchronization of the communication: starting it and stopping it.
- Maintain the status of the communication (in case 1 user "falls" off or the internet connection gets broken) and re-communicating users automatically.
- Take care or routing the communication from user's browsers through out the Internet.

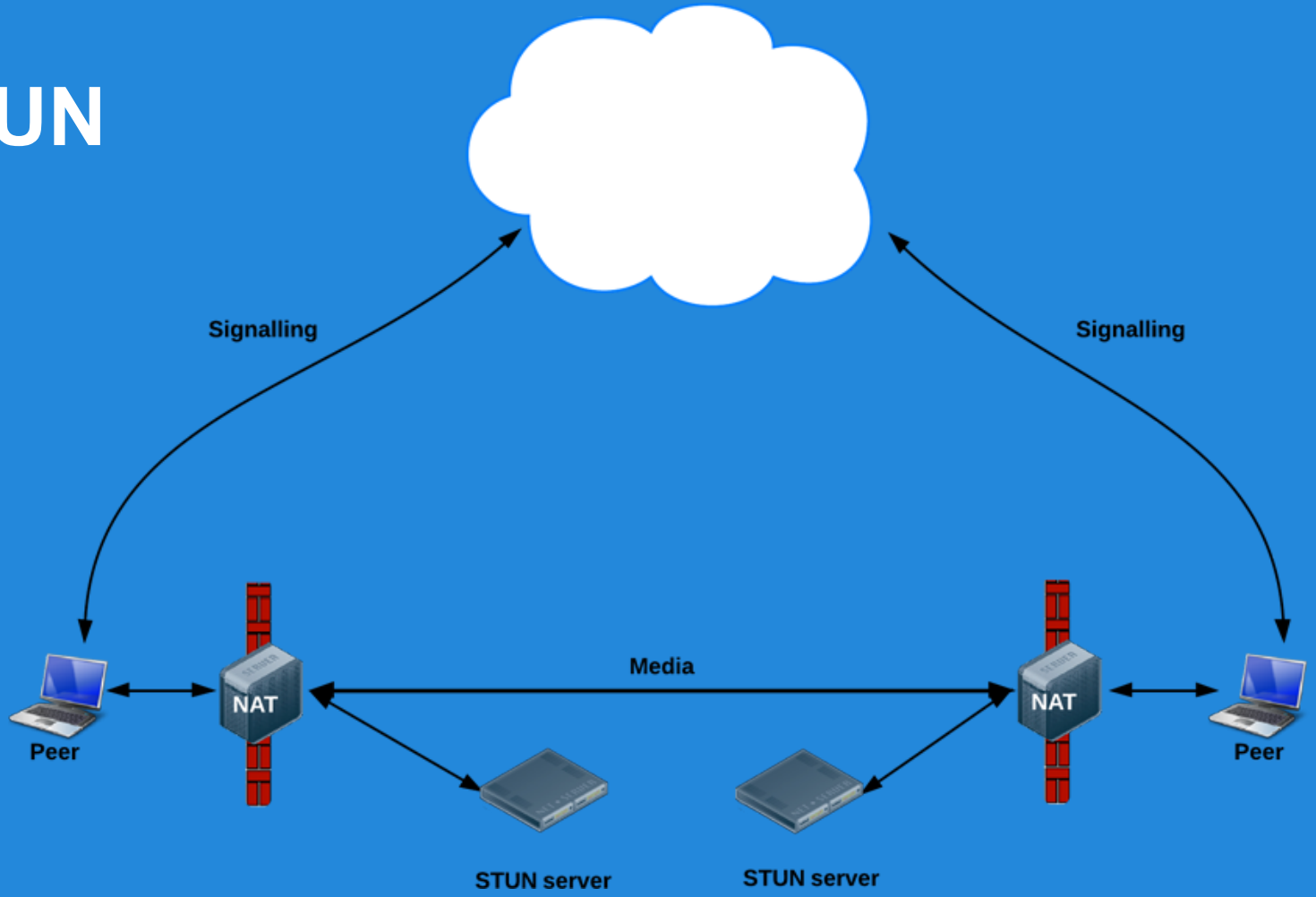
Signaling Diagram



STUN

- Tell me what my public IP address is
- Simple server, cheap to run
- Data flows peer-to-peer

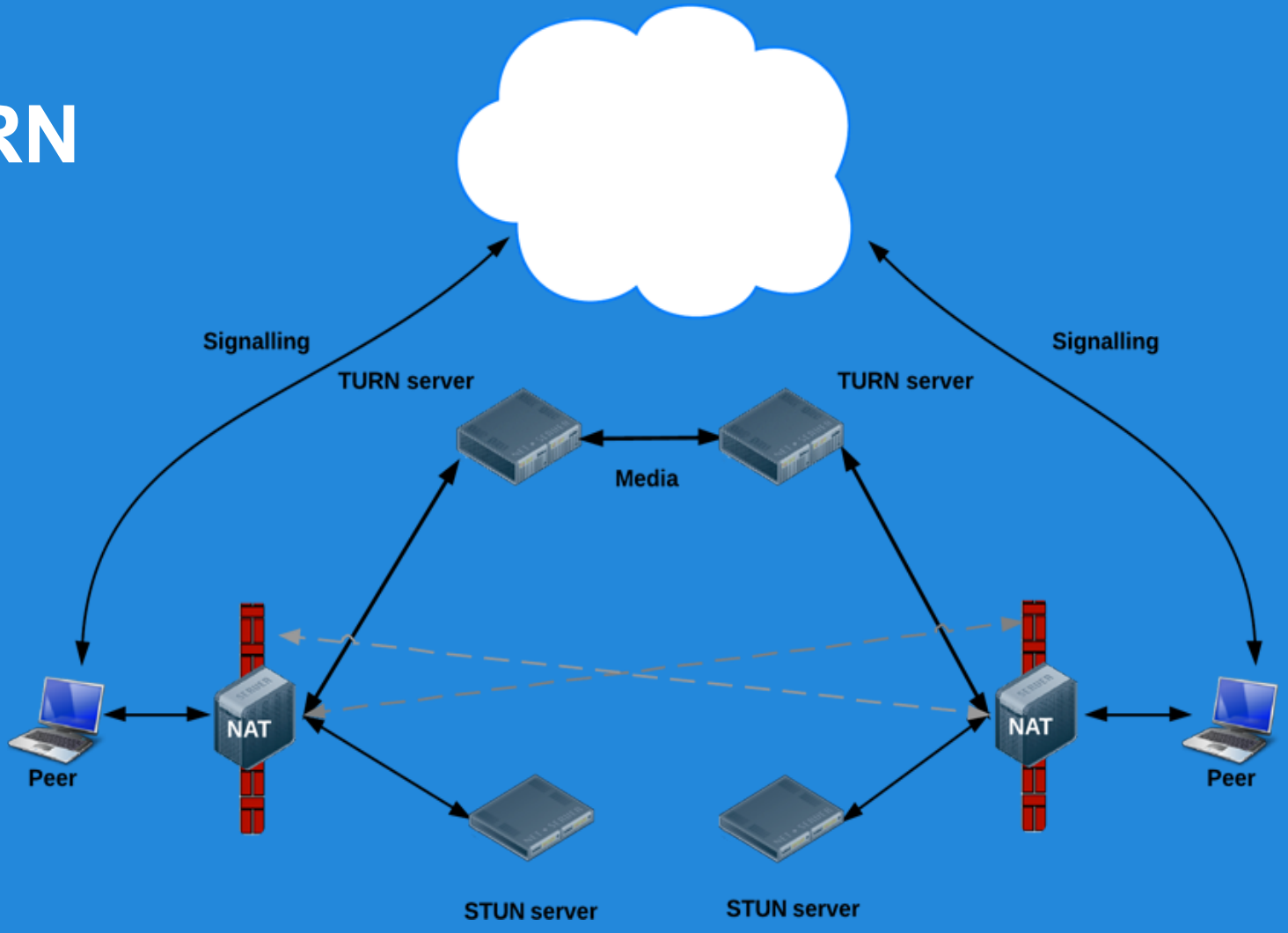
STUN


Signalling
Signalling
Media
NAT
NAT
Peer
Peer
STUN server
STUN server

TURN

- Provide a cloud fallback if peer-to-peer communication fails
- Data is sent through server, uses server bandwidth
- Ensures the call works in almost all environments

TURN


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

Servers - NodeJS & Amazon

Signalling servers implemented with NodeJS

https://bitbucket.org/webrtc/codelab

Signalling servers implemented with MV ready to deploy on Amazon

http://blog.knoldus.com/2013/10/24/configure-turn-server-for-webrtc-on-amazon-ec2/

Servers - references

Signalling server implemented with Socket.io

https://github.com/andyet/signalmaster

Infrastructure

http://www.html5rocks.com/en/tutorials/webrtc/infrastructure/

All steps from JS code to configuring the server

https://bitbucket.org/webrtc/codelab#markdown-header-step-5-set-up-a-signaling-server-and-exchange-messages

Experiments & Docs

Ascii camera

http://idevelop.ro/ascii-camera/

FaceKat

http://shinydemos.com/facekat/

Google I/O Presentation 2013

http://io13webrtc.appspot.com/

GoogleHangout's cousin

https://apprtc.appspot.com/?r=59559604

WebCamToy

http://webcamtoy.com/app/

Today

Socialte.me

Social Network - Meetings & Conferences - Task management - WebRTC with Flash fallback

http://socialite.me/

ShareFest

Share files through Internet…(so it's Bittorrent dude!)

without installing **any** software ;)

https://www.sharefest.me/

https://www.sharefest.me/faq#security


Chuck Norris Approves
Chuck Norris


THANK YOU
FOR YOUR ATTENTION
risovach.ru