Signaling:

- WebRTC can’t create connections without some sort of server in the middle. We call this the signal channel or signaling service.
- Peer A who will be the initiator of the connection, will create an Offer. They will then send this offer to Peer B using the chosen signal channel. Peer B will receive the Offer from the signal channel and create an Answer. They will then send this back to Peer A along the signal channel.

Session descriptions:

- The configuration of an endpoint on a WebRTC connection is called a session description.
- SDP is a short structured textual description.
- It conveys the name and purpose of the session, the media, protocols, codec formats, timing and transport information.
- A tentative participant checks these information and decides whether to join a session and how and when to join a session if it decides to do so.
- The format has entries in the form of <type> = <value>, where the <type> defines a unique session parameter and the <value> provides a specific value for that parameter.
- The general form of a SDP message is –
- x = parameter1 parameter2 ... parameterN

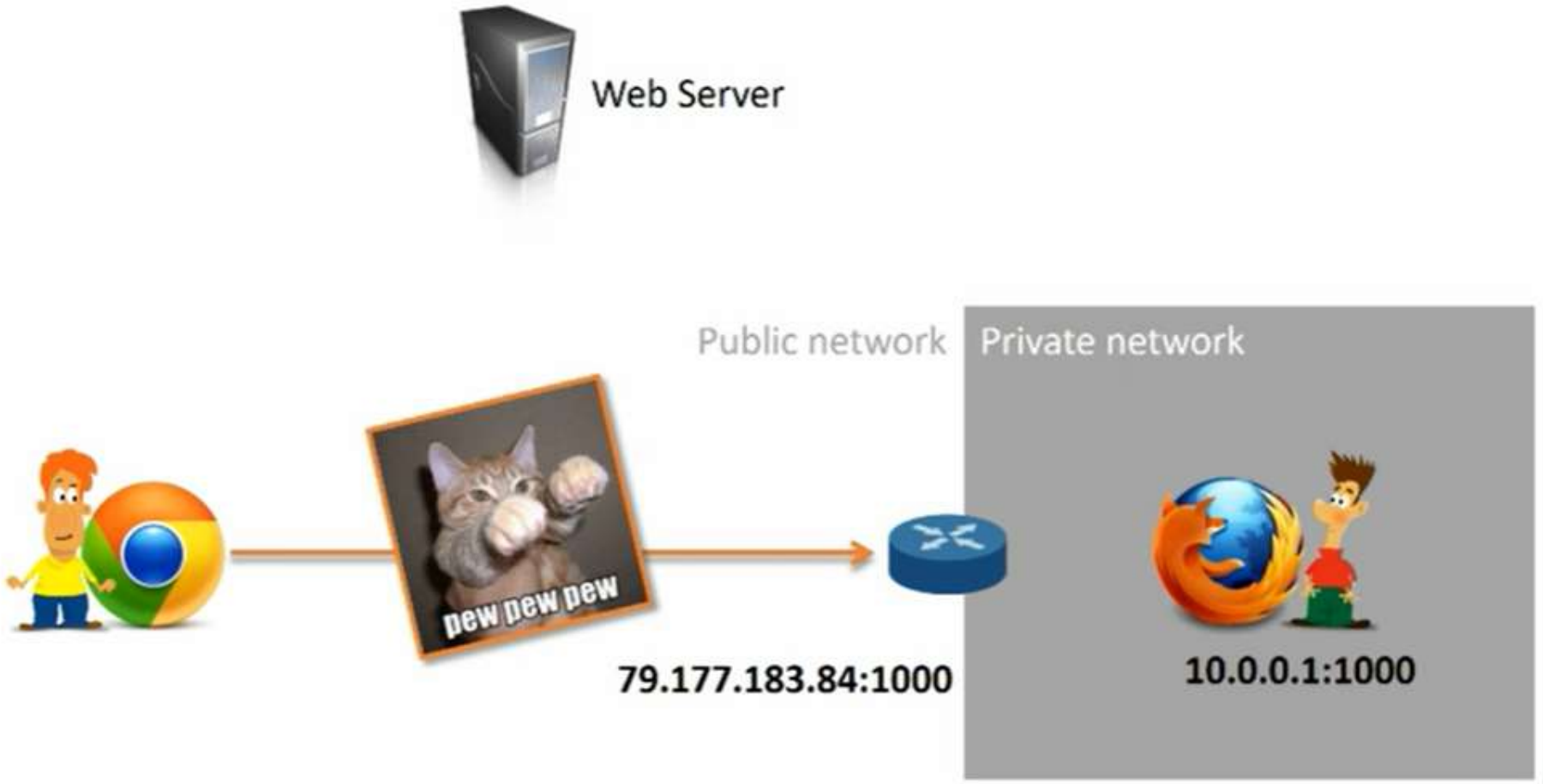
Sending P2P

Web Server

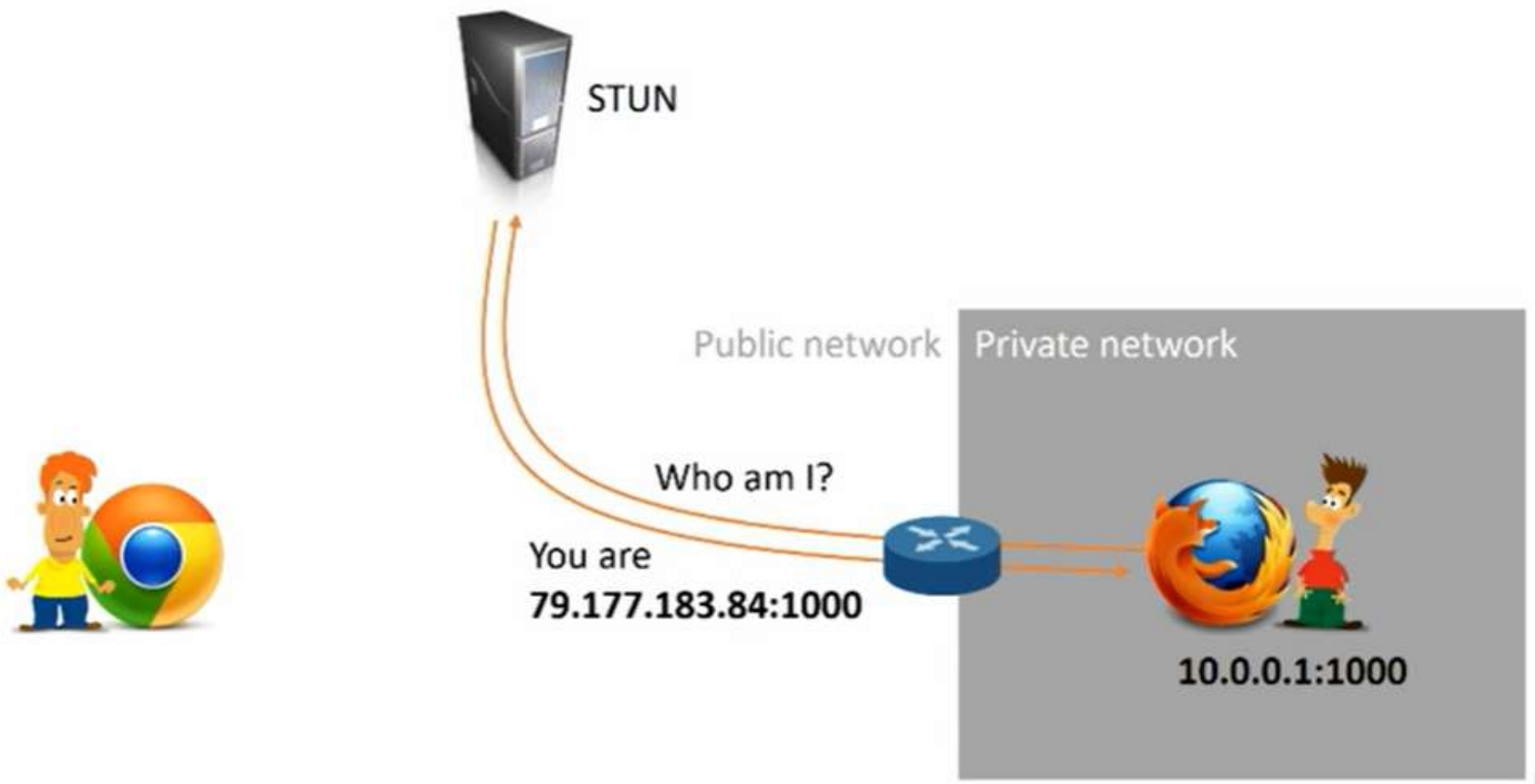
But how do we get there?

Web Server

How can brownhead tell redhead who he is?

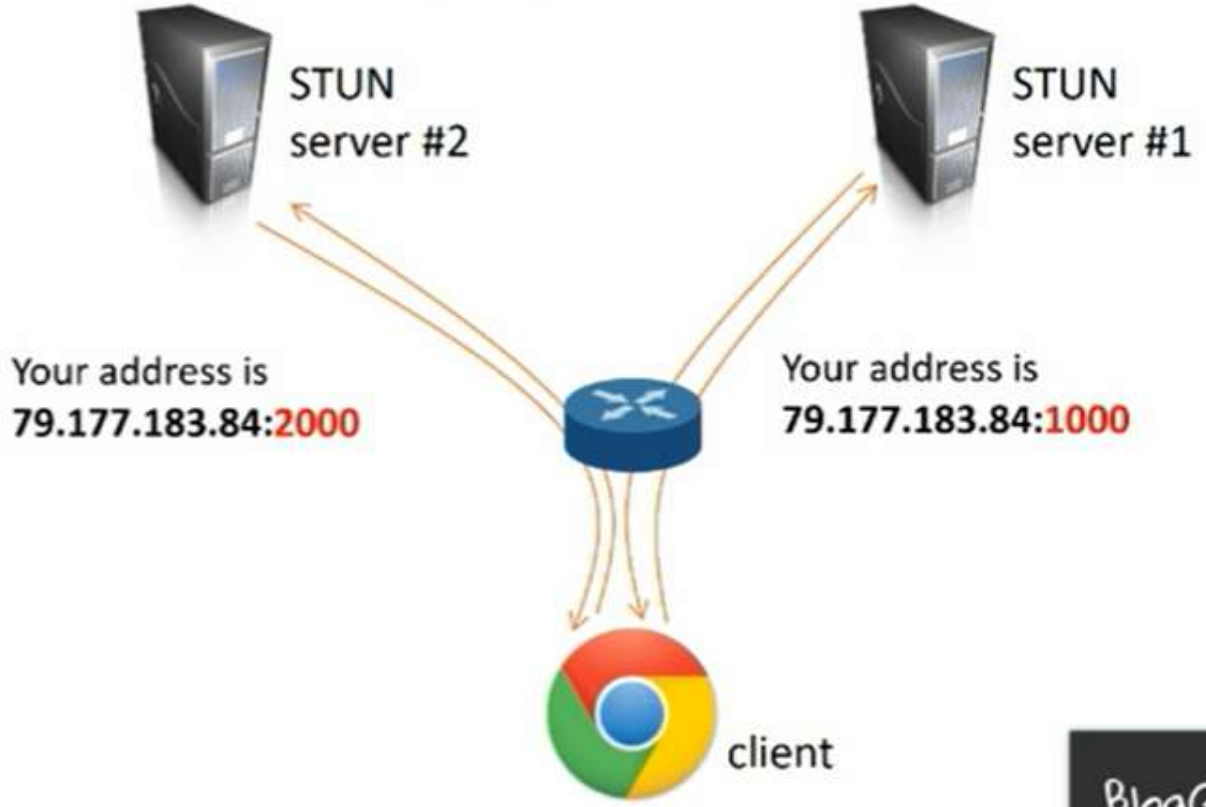


Brownhead can ask!



Sometime, it isn't enough though

- Not all NATs are created equal
- Symmetric NATs are a real pain: mapping is done by both source and destination IP addresses



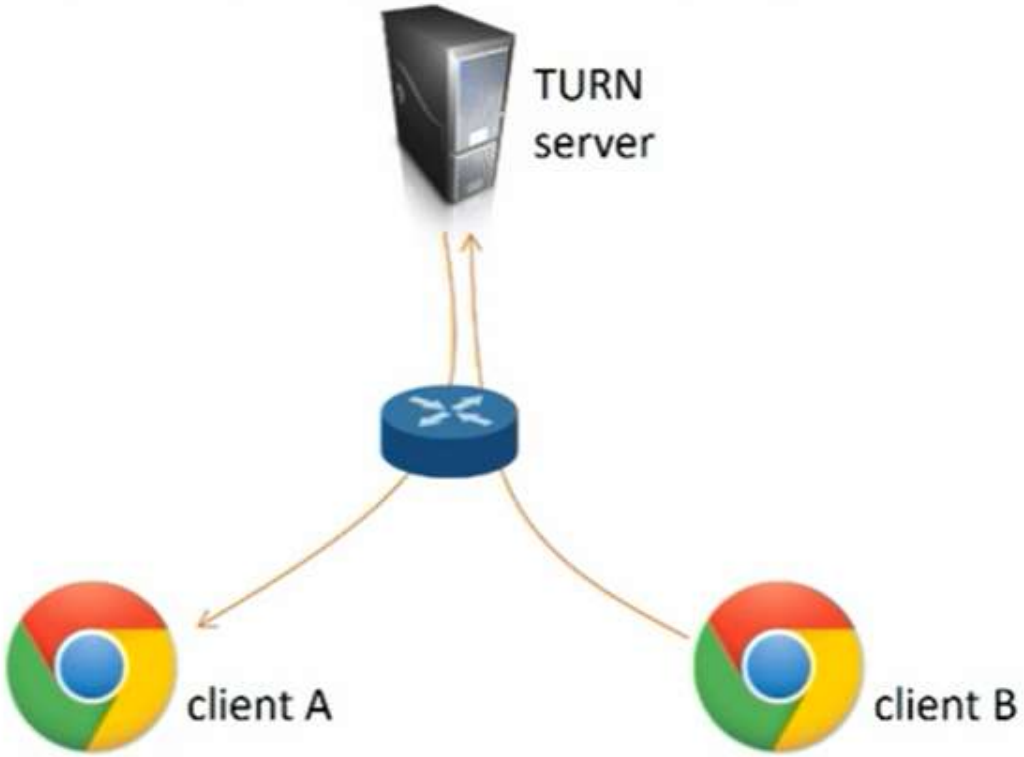
So we TURN

TURN = Traversal Using Relays around NAT

Why send direct when we can relay everything via a third party?

Comes at a price:

- Eats up bandwidth
- Eats up servers and CPU
- Adds latency
- More moving parts



Comparing STUN vs. TURN

	STUN Server	TURN Server
Purpose	Returns an external IP address	Relays media
Frequency of use	Almost always	Sometimes
Operating costs	Inexpensive	Expensive
Quality impact	No	Possible

2 options then:

1. STUN
   - Easy on backend resources
   - But won't always work
2. TURN
   - Works almost always
   - But eats up our backend resources

We use ICE to decide

Resolve connectivity issues by conducting connectivity checks

Client collects addresses of possible candidates

- Host candidates (local addresses)
- Server reflexive candidates (obtained via STUN servers)
- Relayed candidates (TURN server addresses)

Client sends candidates via SDP

Client tries to connect to received candidates until it succeeds