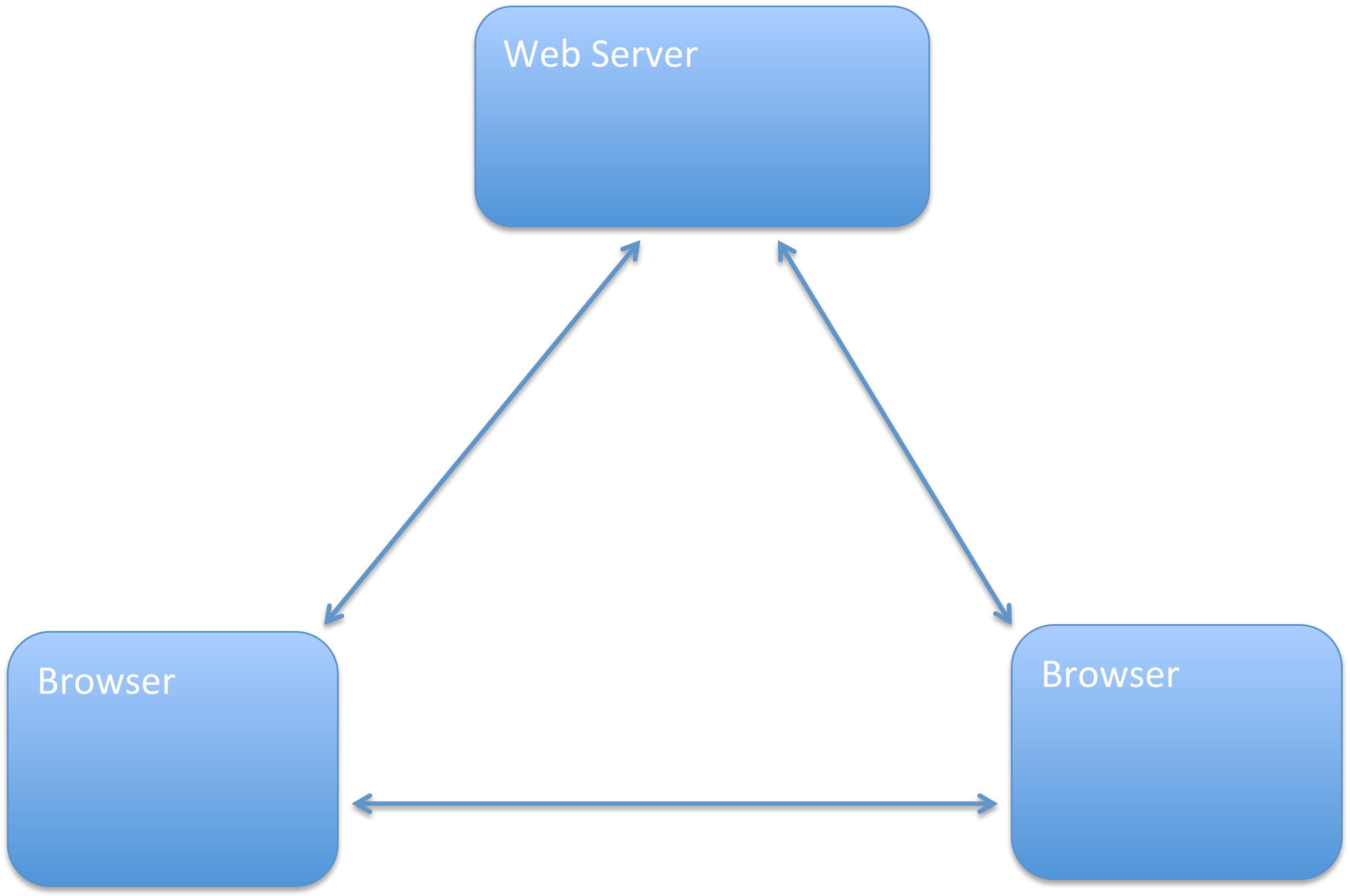
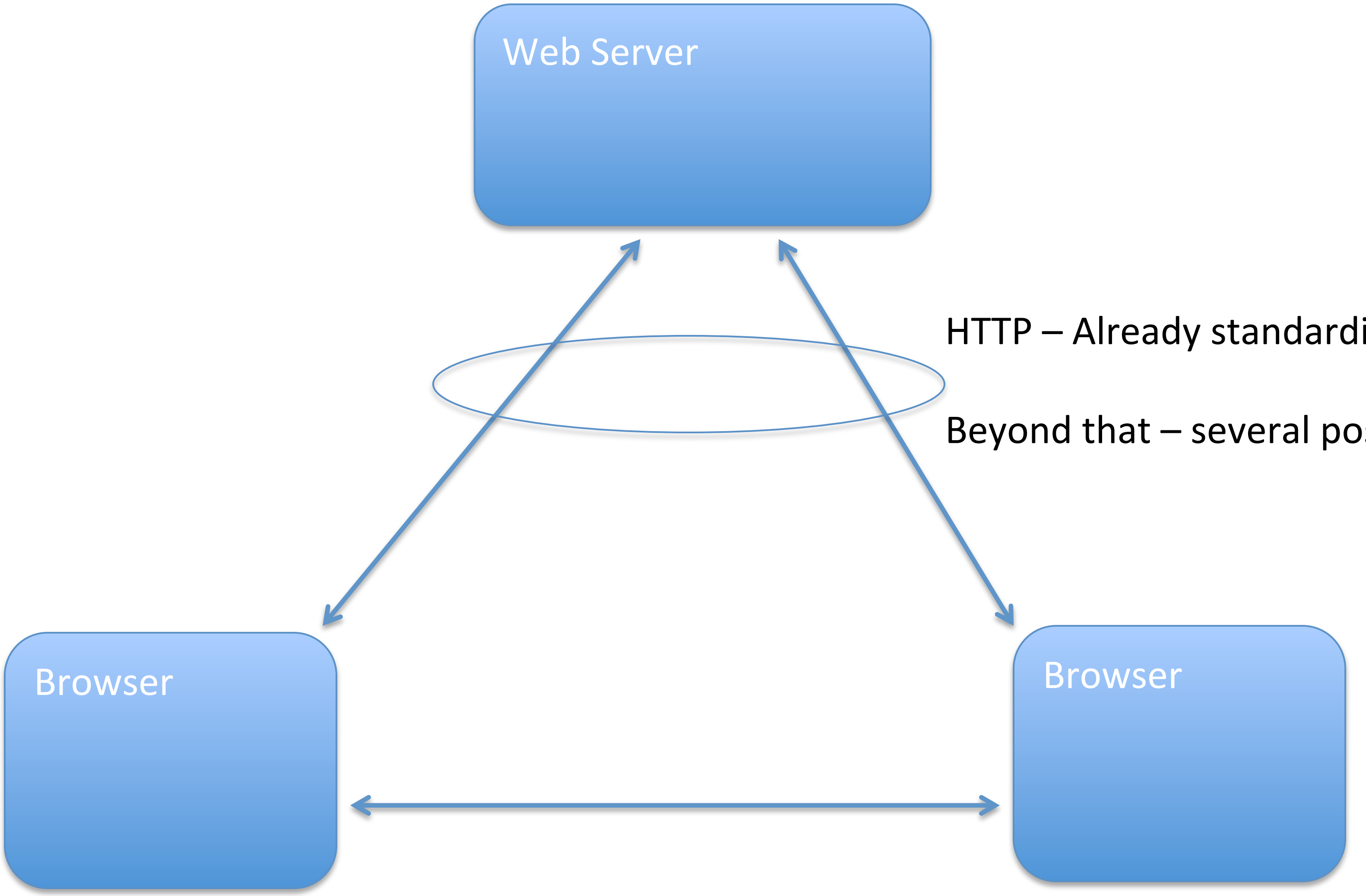
RTCWEB Signaling

Matthew Kaufman

Scope



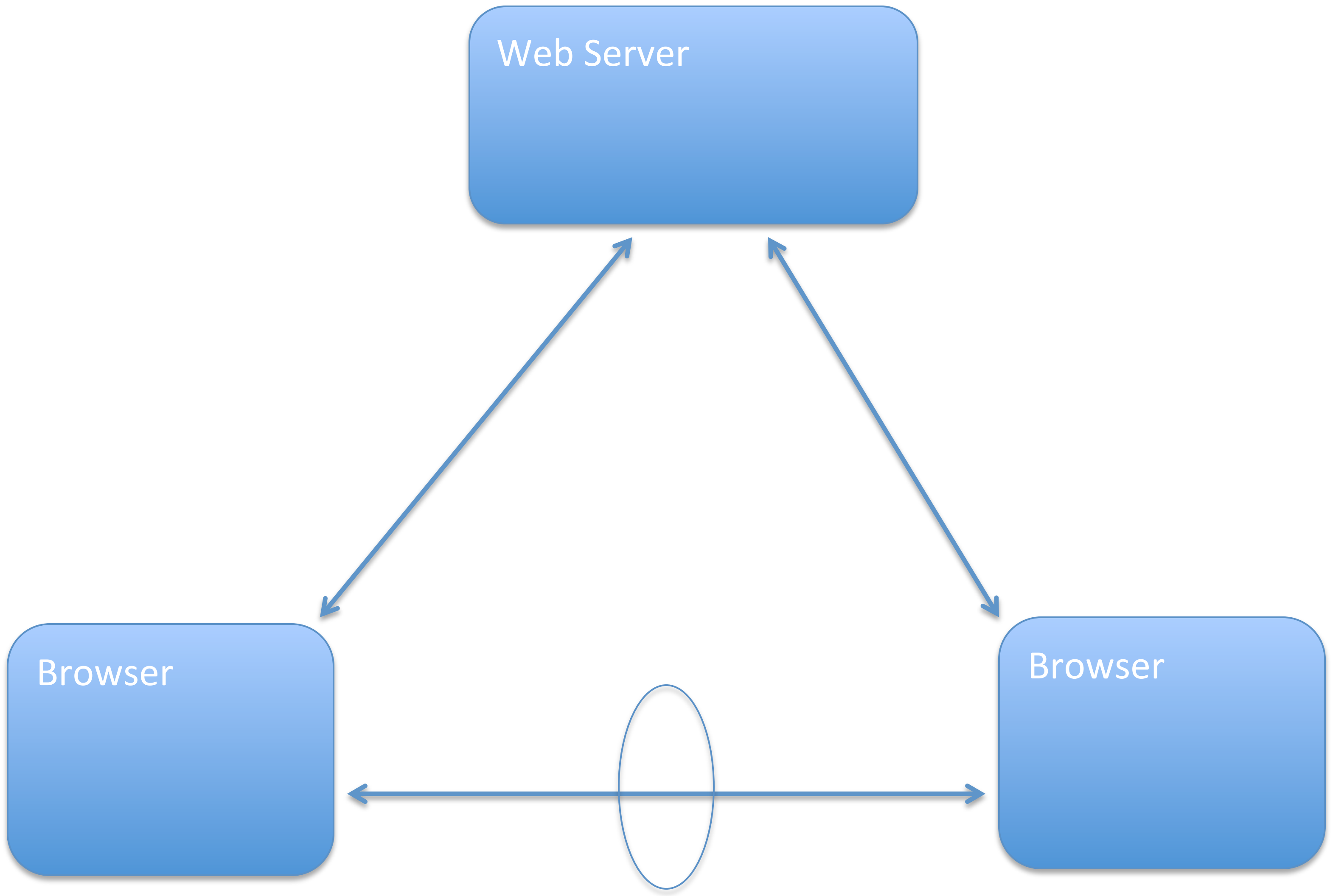
Scope



HTTP – Already standardized

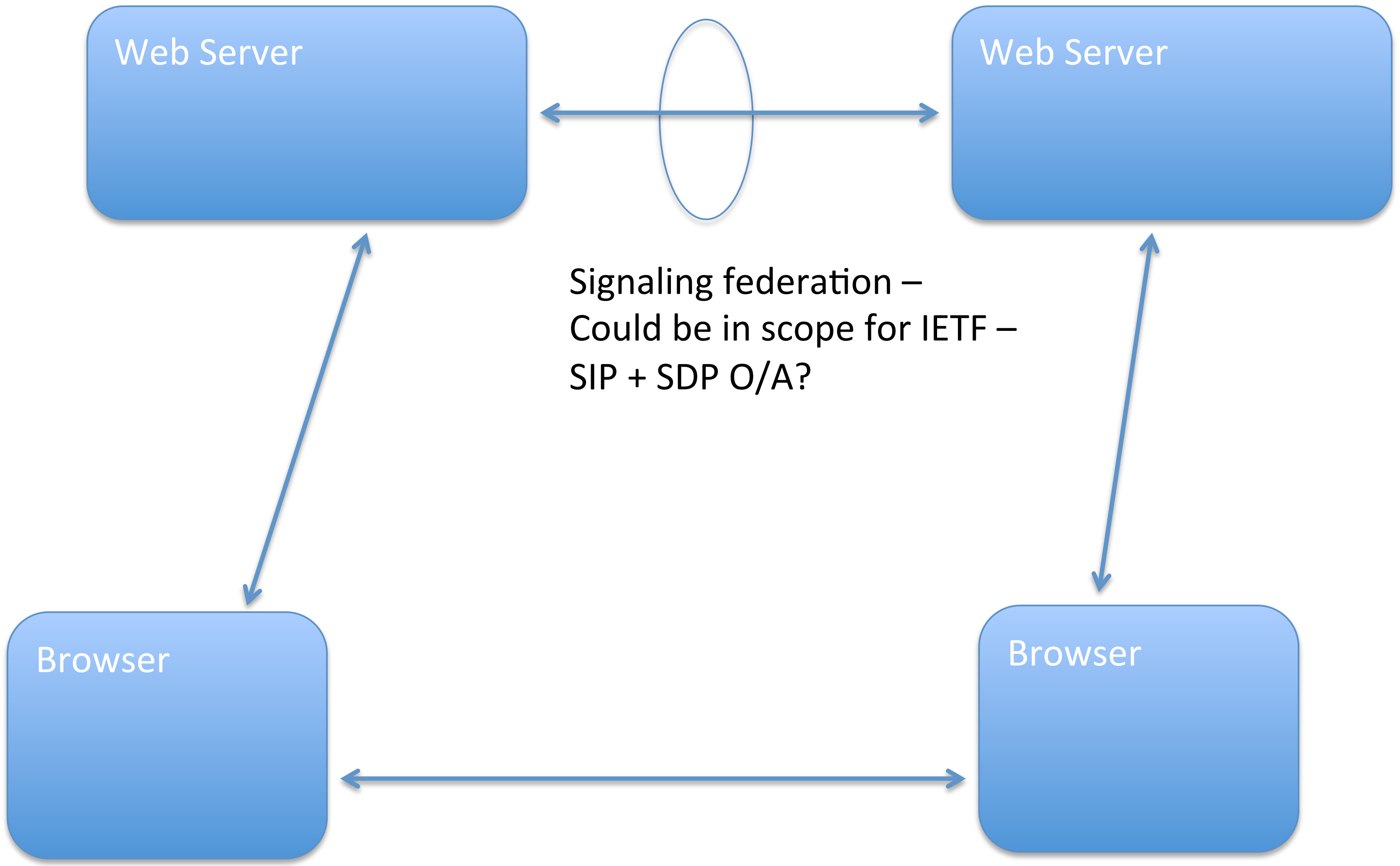
Beyond that – several possibilities

Scope



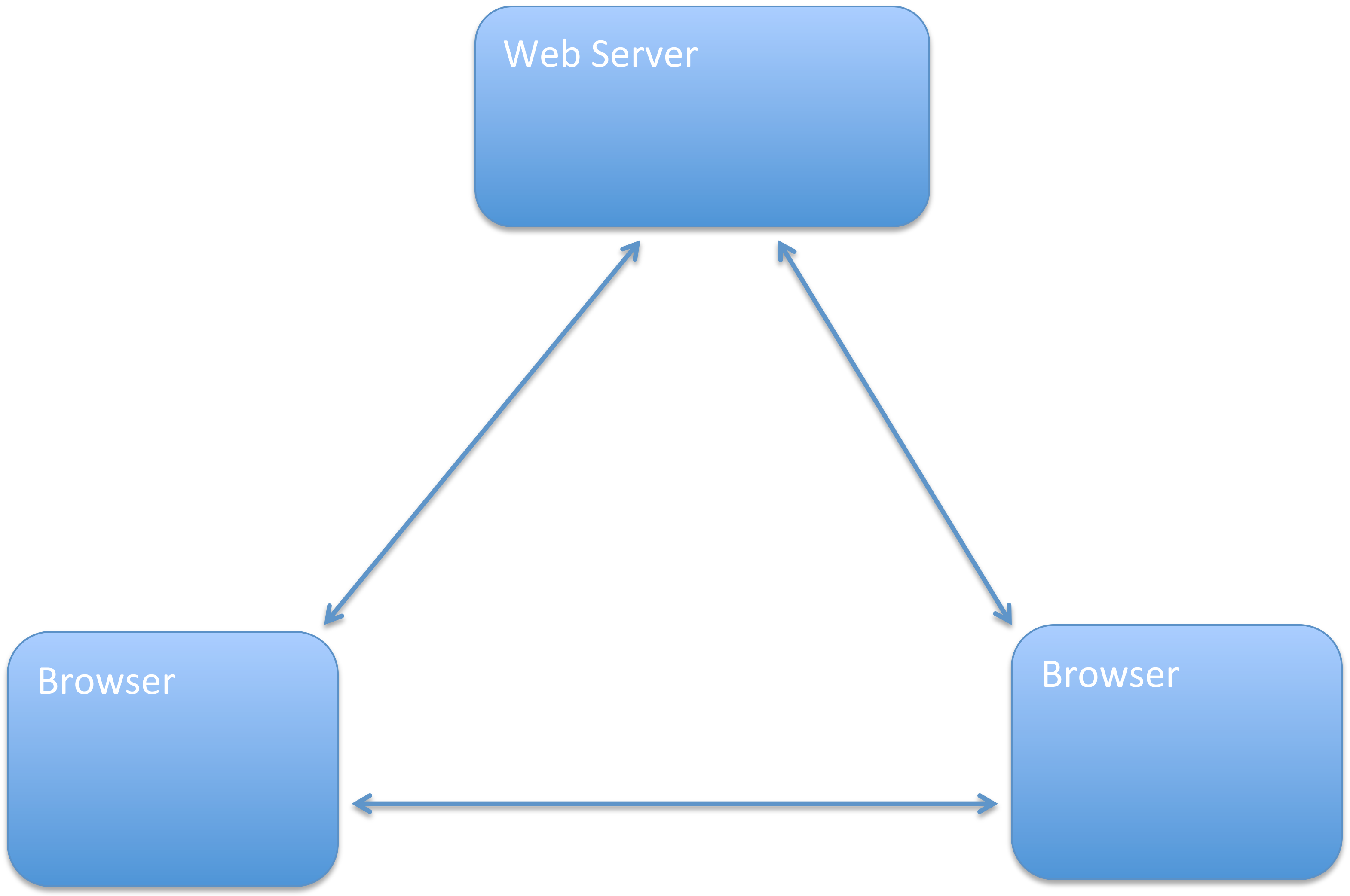
Media Transport – In scope for IETF – DTLS-SRTP + ICE?

Scope

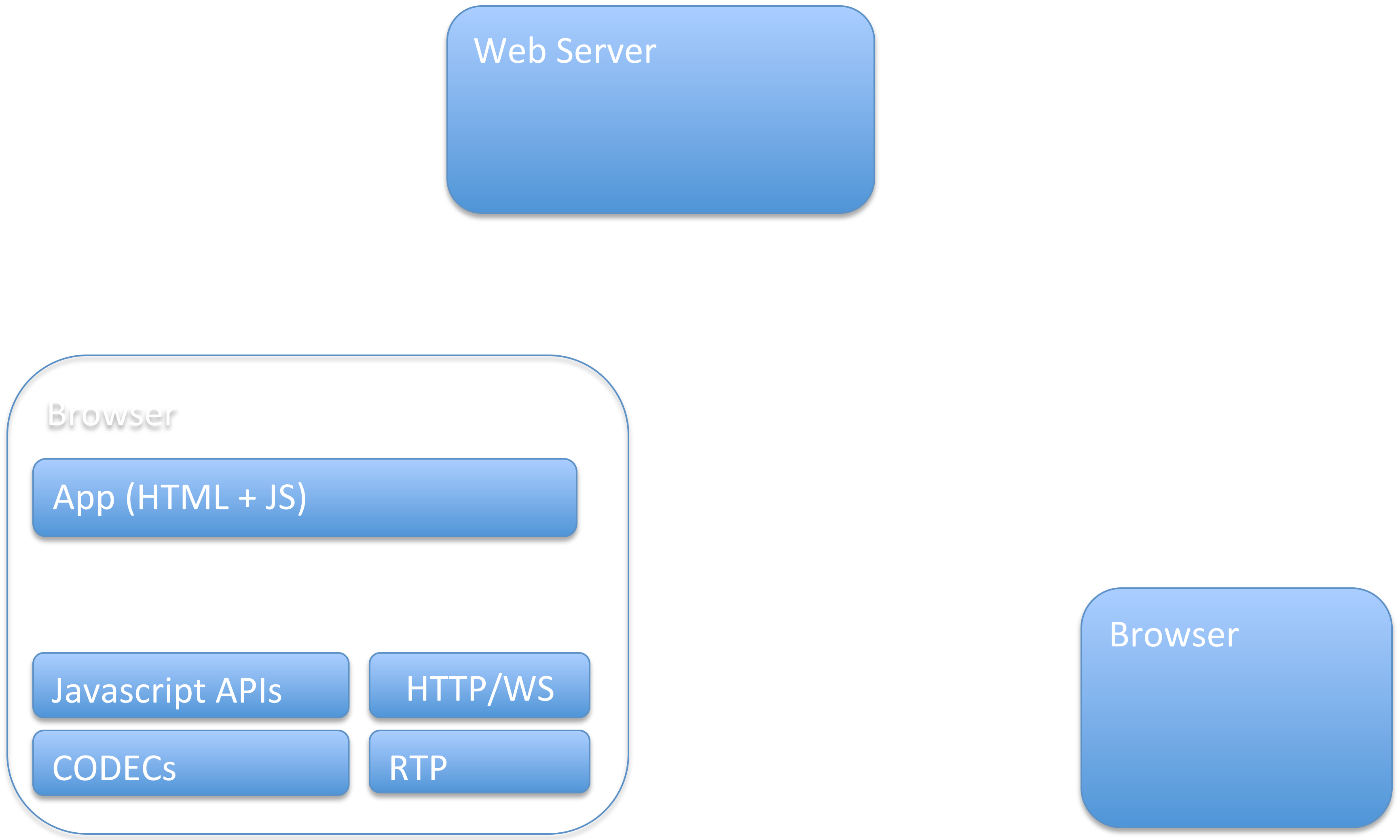


Signaling federation –
Could be in scope for IETF –
SIP + SDP O/A?

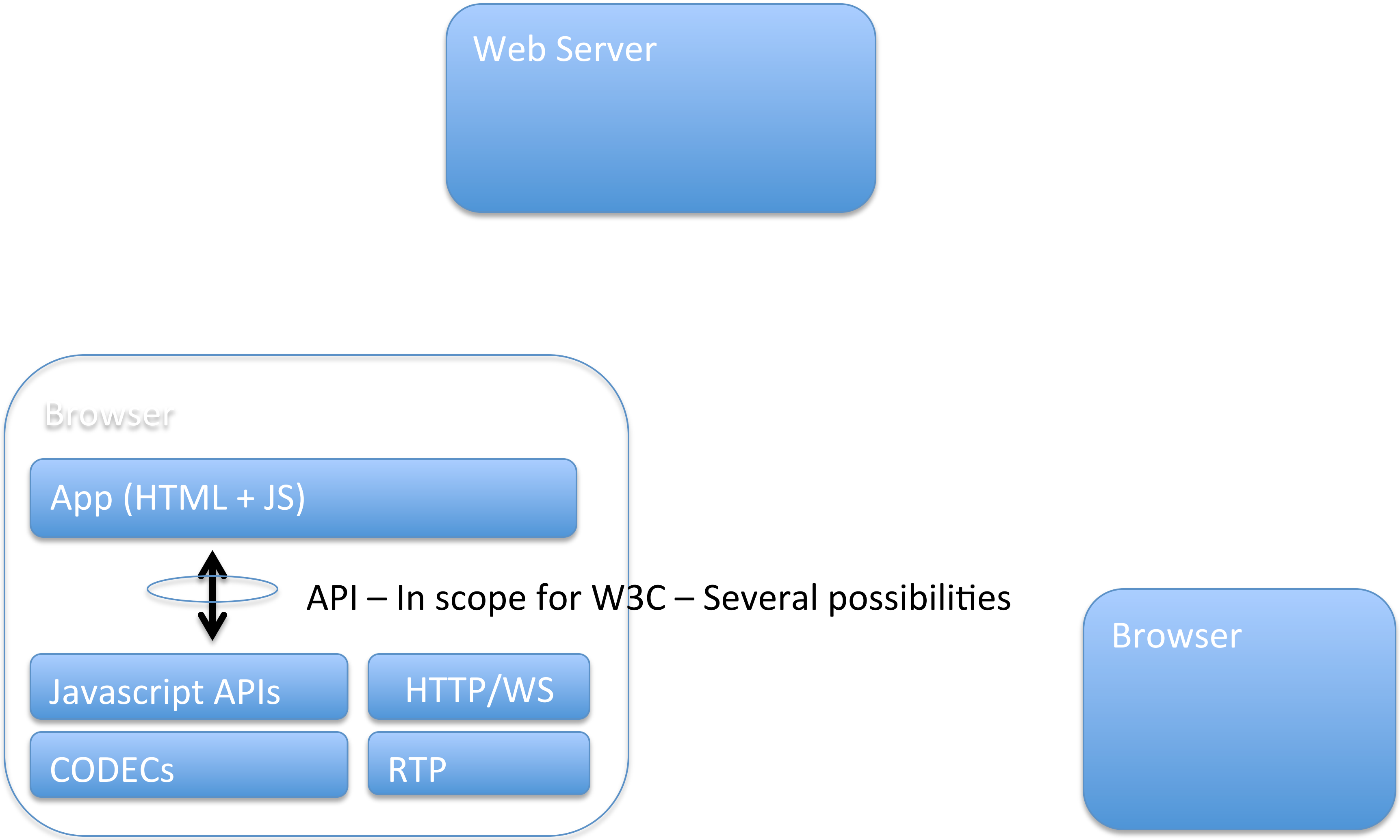
Scope



Scope


Web Server
Browser
App (HTML + JS)
Javascript APIs
HTTP/WS
CODECs
RTP
Browser

Scope


Web Server
Browser
App (HTML + JS)
API – In scope for W3C – Several possibilities
Javascript APIs
HTTP/WS
CODECs
RTP
Browser

Identified In-Scope Questions

1. Should the format of what travels over HTTP between the web server and the browser be standardized?
2. What should the media transport protocol be?
3. Should the format of what travels between two federated web servers be standardized?
4. What should the Javascript API be like?

Question 1

- Should the format of what travels over HTTP between the web server and the browser be standardized?
  - Choices:
    - No, leave it up to the application developer
    - Yes, it should be SIP
    - Yes, it should be a lightweight SIP
    - Yes, it should be not SIP but use SDP Offer/Answer

Question 1

- No, leave it up to the application developer
  - Maximum flexibility
  - Works for RTC applications other than building phones
- Yes, it should be SIP
  - Very high implementation bar, and still missing functionality
- Yes, it should be a lightweight SIP
  - Slippery slope (early media? prack? redirect? Etc?)
  - Not actually compatible with existing servers/services
- Yes, it should be not SIP but use SDP Offer/Answer
  - Does make answering Question 3 (and implementing federation) easier.
  - Does not support cases where early determination of capabilities is useful
  - Does not work well for some multiparty and other "non-phone" use cases
  - Not clear why the IETF or W3C should be telling Javascript programmers what they can send over XMLHTTPRequest anyway

Question 2

- What should the media transport protocol be?
  - Not in scope for these slides, but I think we can solve this with existing IETF protocols +/- minor changes

Question 3

- Should the format of what travels between two federated web servers be standardized?
  - Choices:
    - Yes, it should be SIP
    - Yes, it should be something else
    - No, that's up to the two website operators to work out

Question 3

- Yes, it should be SIP + SDP O/A
    - There are good arguments for this
    - This is probably an area that IETF should explore **later**
- Yes, it should be something else
    - The arguments for this are weaker
- No, that's up to the two website operators to work out
    - They will probably choose to use SIP and SDP O/A anyway
- But this question is NOT the same as Question 1
    - If we need SDP O/A for federation, it doesn't necessarily follow that SDP O/A must be used between the browser and its web site

Question 4

- What should the Javascript API be like?
  - Before answering: Is this really an IETF problem, or is this a W3C problem?
- Choices are on multiple axes
  - How much of calling is “built in”?
    - How does address selection and NAT traversal work?
    - How are CODECs selected?

Question 4

- How much of calling is “built in”?
  - One extreme:
    - phone = new SIPPhone();
    - phone.call(“sipuser@example.com”);
    - This would then imply that SIP travels over the wire to the servers (maybe not even over HTTP!) and SDP offer/answer is used
    - Not much room for innovation here

Question 4

- How much of calling is "built in"?
  - Other extreme:
    - Javascript developer gets a PeerConnection object
    - Runs an ICE implementation in Javascript using calls that allow for sending and receiving of STUN probes
    - Gets a camera object and a codec object and examines the capabilities
    - Negotiates the capabilities with the web server using a proprietary protocol
    - Explicitly sets the camera and codec properties
    - Attaches the codec to the PeerConnection and starts sending
    - Lots of interesting (though perhaps difficult) Javascript code to write and new applications to build

Question 4

- How much of calling is “built in”?
  - And of course many intermediate choices

Question 4

- How does address selection and NAT traversal work?
  - Option A
    - PeerConnection is told what to connect to by being passed a blob of SDP
    - PeerConnection runs ICE, generates more blobs of SDP containing ICE candidates as events, these are transported to the other end and pushed into the PeerConnection

Question 4

- How does address selection and NAT traversal work?
  - Option B
    - PeerConnection is told what to connect to by being passed a candidate address or list of candidate addresses
    - PeerConnection runs ICE, generates blobs of SDP (or some other format) containing ICE candidates as events, these are transported to the other end and pushed into the PeerConnection

Question 4

- How does address selection and NAT traversal work?
  - Option C
    - PeerConnection has APIs that let the developer implement ICE or any equivalent algorithm that meets the “STUN probe is used to prove far end wants to receive media” security requirement

Question 4

- How does address selection and NAT traversal work?
  - This is discussed in draft-cbran-rtcweb-nat-01

Question 4

- How are CODECs selected?
  - SDP offer/answer is used between the PeerConnection objects at each end over their event-based signaling channel and the results determine the CODECs in use
  - Direct Javascript APIs allow the developer to query for capabilities and then select a CODEC and specify necessary parameters

Question 4

- SDP offer/answer is used between the PeerConnection objects at each end over their event-based signaling channel and the results determine the CODECs in use
    - Tempting, as you already need this channel for ICE
    - Actually, the Javascript can manipulate the SDP that comes out and even parse it and convert it to another form… so this doesn't mandate "sending SDP over the wire" (Question 1)
    - But it would probably encourage developers to simply pass the SDP intact from end to end

Question 4

- Direct Javascript APIs allow the developer to query for capabilities and then select a CODEC and specify necessary parameters
  - Allows for the building of applications which query for capabilities before offering calling to the user
  - Allows for more complex multi-party calling (know in advance which codec one could switch everyone else to in order to support a new joiner with a more limited set)
  - Leverages the Device APIs that are coming into existence

Question 4

- Direct Javascript APIs allow the developer to query for capabilities and then select a CODEC and specify necessary parameters
    - STILL ALLOWS for SDP Offer/Answer on the wire, simply by writing some Javascript to create it
    - STILL ALLOWS for SDP Offer/Answer in the federation case, by creating SDP at the browser in Javascript or up at the server

Question 4

- Direct Javascript APIs allow the developer to query for capabilities and then select a CODEC and specify necessary parameters
  - COULD STILL LEVERAGE the MMUSIC work performed for each new CODEC by exposing the capabilities and parameter setting as compatible strings, rather than individual Javascript knobs
  - COULD EVEN use SDP, but not as SDP offer/answer, instead as a command mechanism as in other protocols like MGCP

Recommendations

- Try to avoid solving problems that are out of scope
  - We have enough that are in scope already
- Try to maximize the flexibility by turning the browser into the next operating system in which we create RTC applications, not by bolting a phone on the side

Therefore

- Question 1: No. Up to the app developer, just as in web email clients
- Question 2: Probably DTLS-SRTP, possibly with multiplexing added, and ICE for NAT traversal
- Question 3: Leave this up to the site operators for now and revisit whether additional SIP (or other) work is needed at a later time
- Question 4: Don't build a SIP phone into the browser
    - Implement ICE in Javascript if practical, otherwise implement ICE natively and pass the ICE candidates in and out via strings in a reasonable format (perhaps even SDP)
    - Don't build SDP offer/answer into the PeerConnection object (where it doesn't belong anyway) for CODEC selection, instead expose APIs for querying capabilities and setting CODEC and CODEC parameters. If possible, leverage types and parameter serialization created by MMUSIC.