WebRTC

WebRTC: Real-Time Communications on the Web

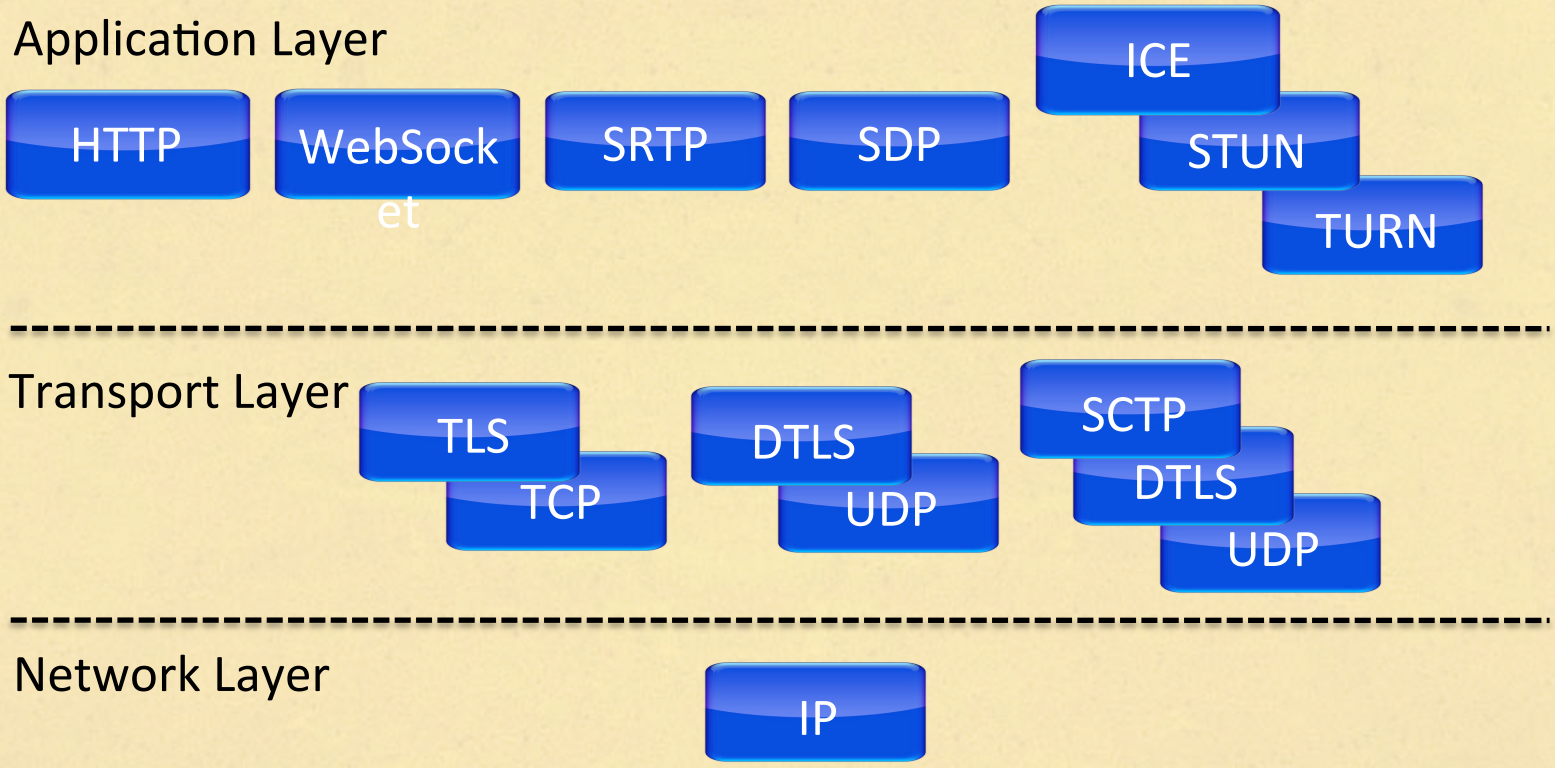
Dan Burnett, PhD
http://standardsplay.com
http://allthingsrtc.org

Alan Johnston, PhD
Rowan University

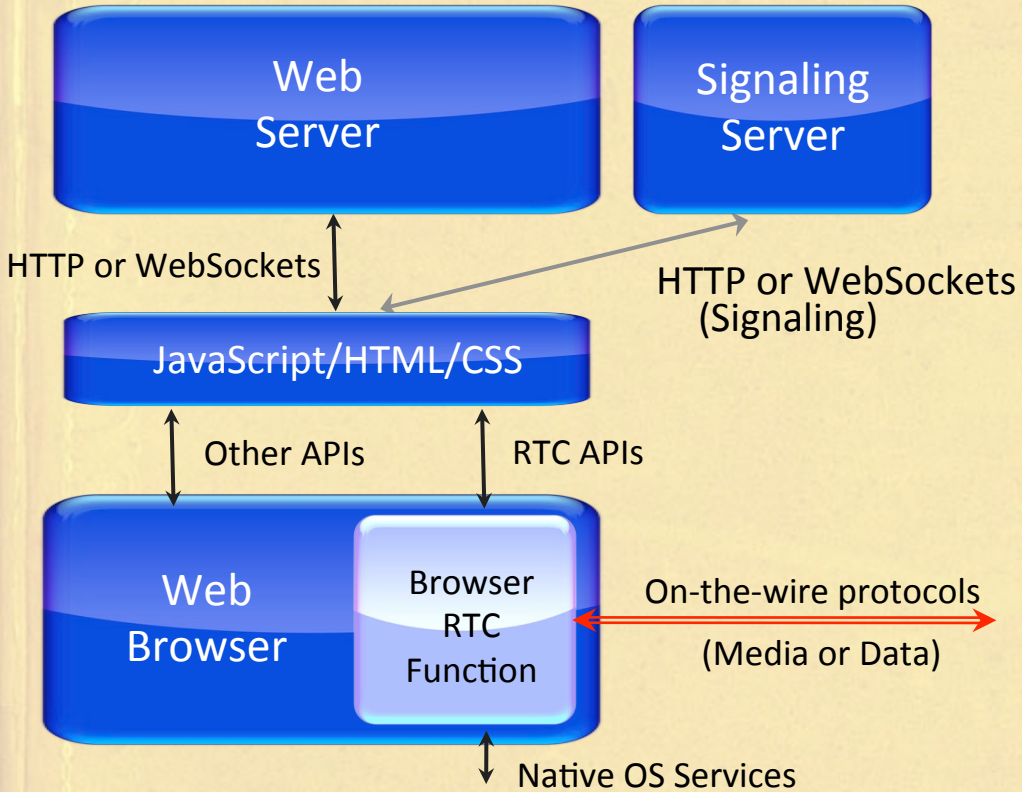
WebRTC: A Joint Standards Effort

- Internet Engineering Task Force (IETF) and World Wide Web Consortium (W3C) are working together on WebRTC
- IETF
  - Protocols – "bits on wire"
  - Main protocols are already RFCs, but many extensions in progress
  - RTCWEB (Real-Time Communications on the Web) Working Group is the main focus, but other WGs involved as well
    - MMUSIC WG (ICE and SDP extensions)
    - TRAM WG (STUN and TURN extensions)
    - AVTCORE WG (RTP extensions)
    - RMCAT WG (RTP Media Congestion Avoidance)
- W3C
  - APIs – used by JavaScript code in HTML5
  - WEBRTC WG and Media Capture TF are main groups

WebRTC Protocols



The Browser RTC Function



- New Real-Time Communication (RTC) Function built-in to browsers
- Contains
  - Audio and video codecs
  - Ability to negotiate peer-to-peer connections
  - Echo cancellation, packet loss concealment
- In Chrome, Firefox, Microsoft EDGE, and Safari today

Benefits of WebRTC

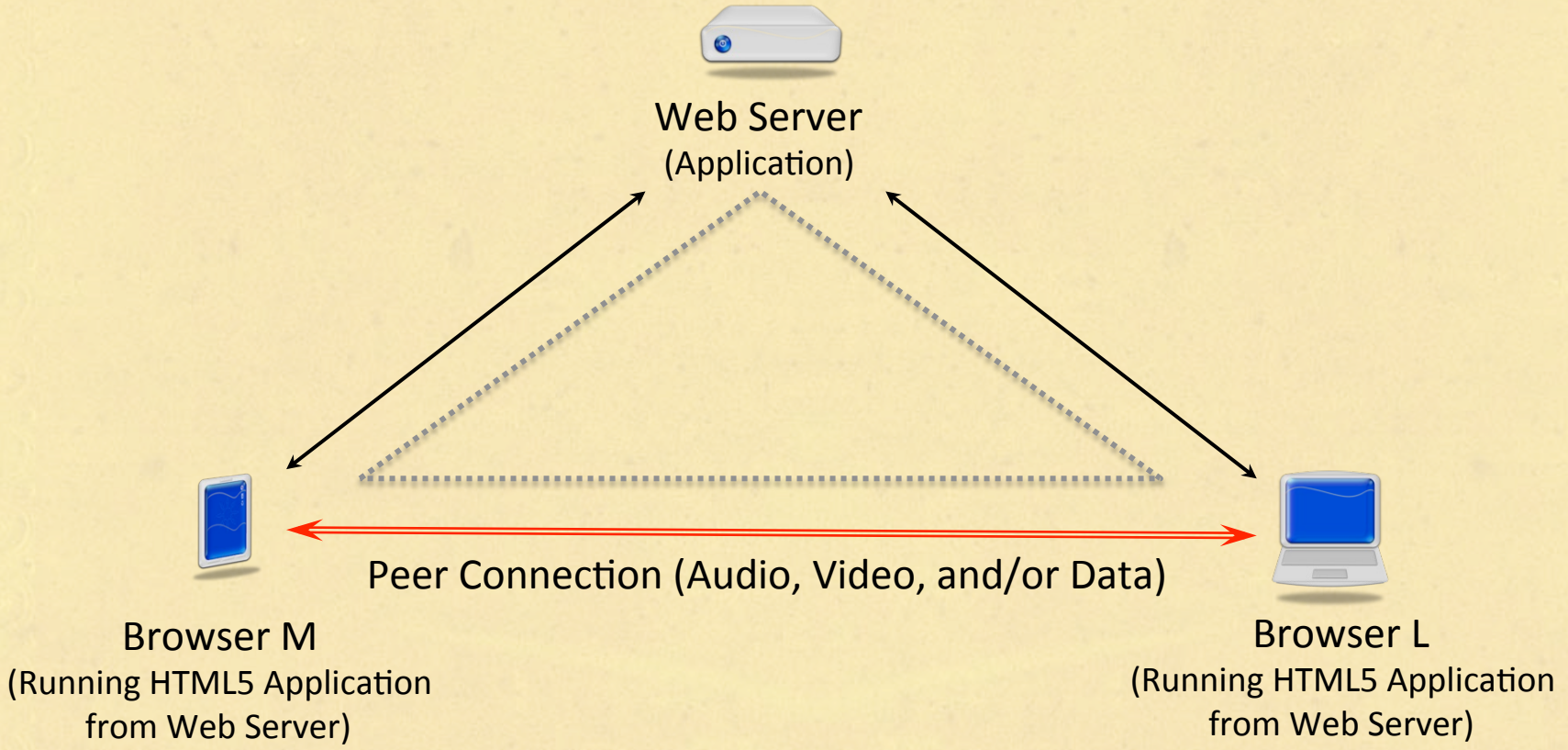
For Developer

- Streamlined development – one platform
- NAT traversal only uses expensive relays when no other choice
- Advanced voice and video codecs without licensing

For User

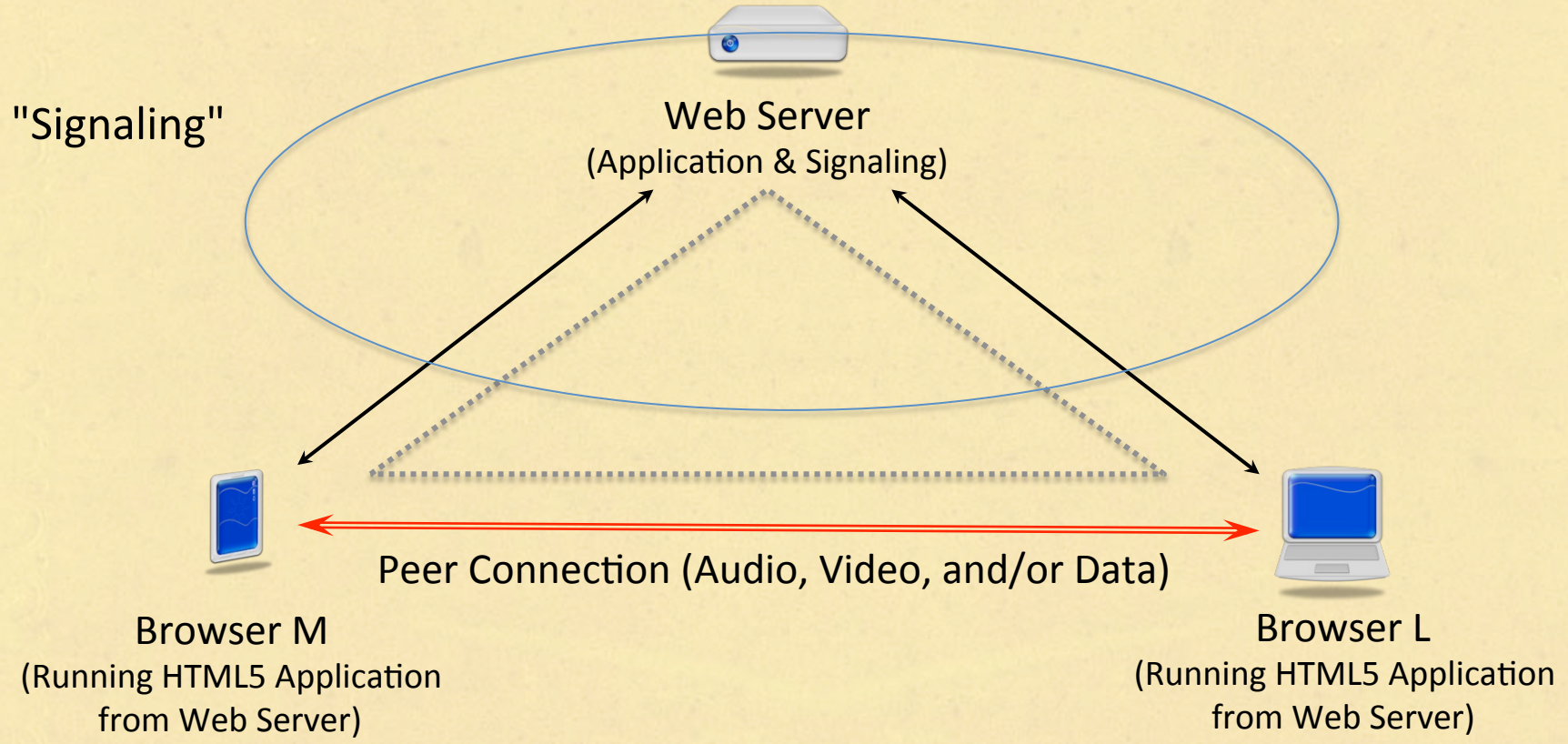
- No download or install – easy to use
- All communicaton encrypted – private
- Reliable session establishment – “just works”
- Excellent voice and video quality
- Many more choices for real-time communication

WebRTC Triangle



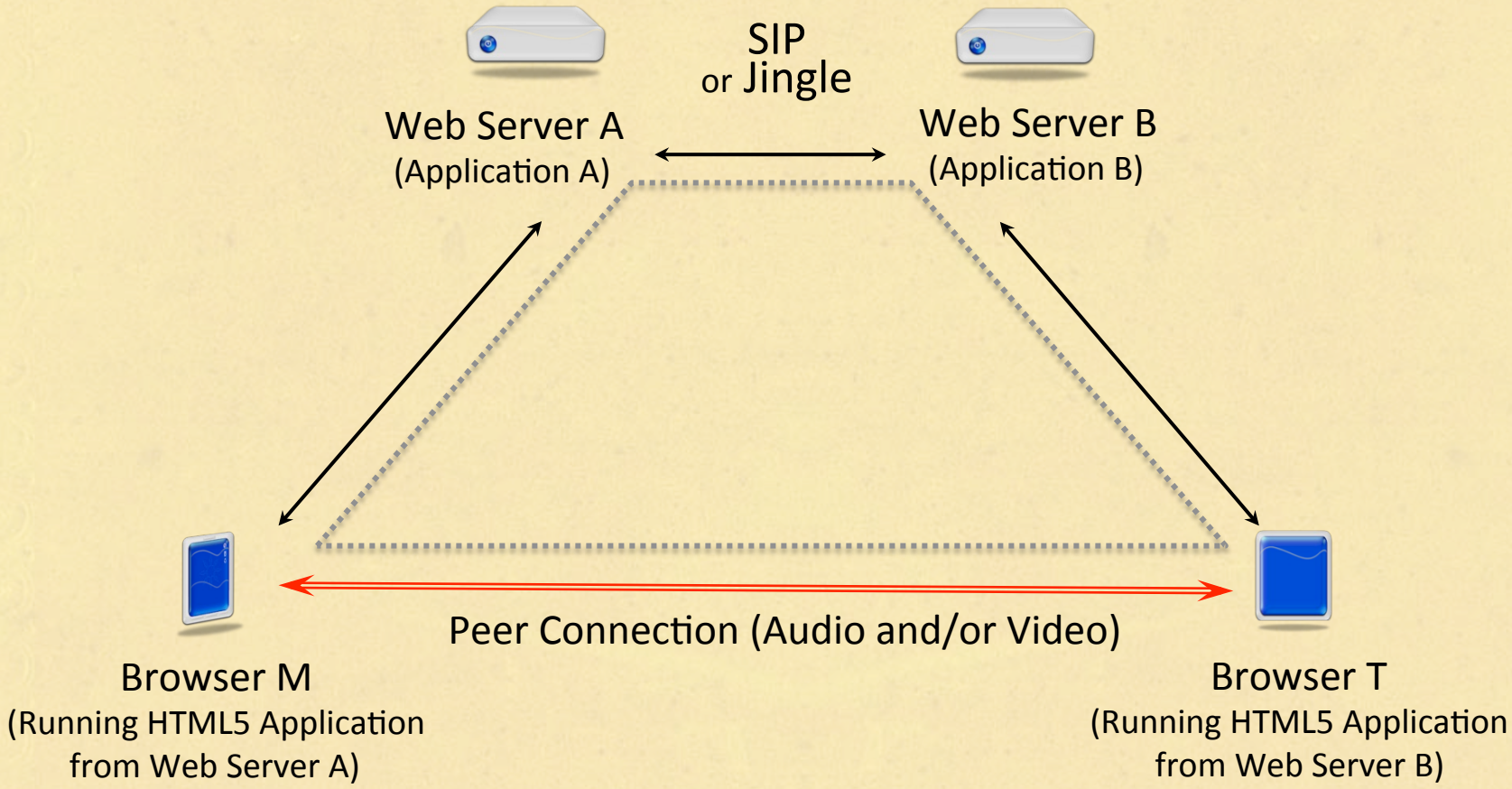
- Both browsers running the same web application from web server
- Peer Connection established between them with the help of the web server

WebRTC Triangle



- Both browsers running the same web application from web server
- Peer Connection established between them with the help of the web server ("Signaling")

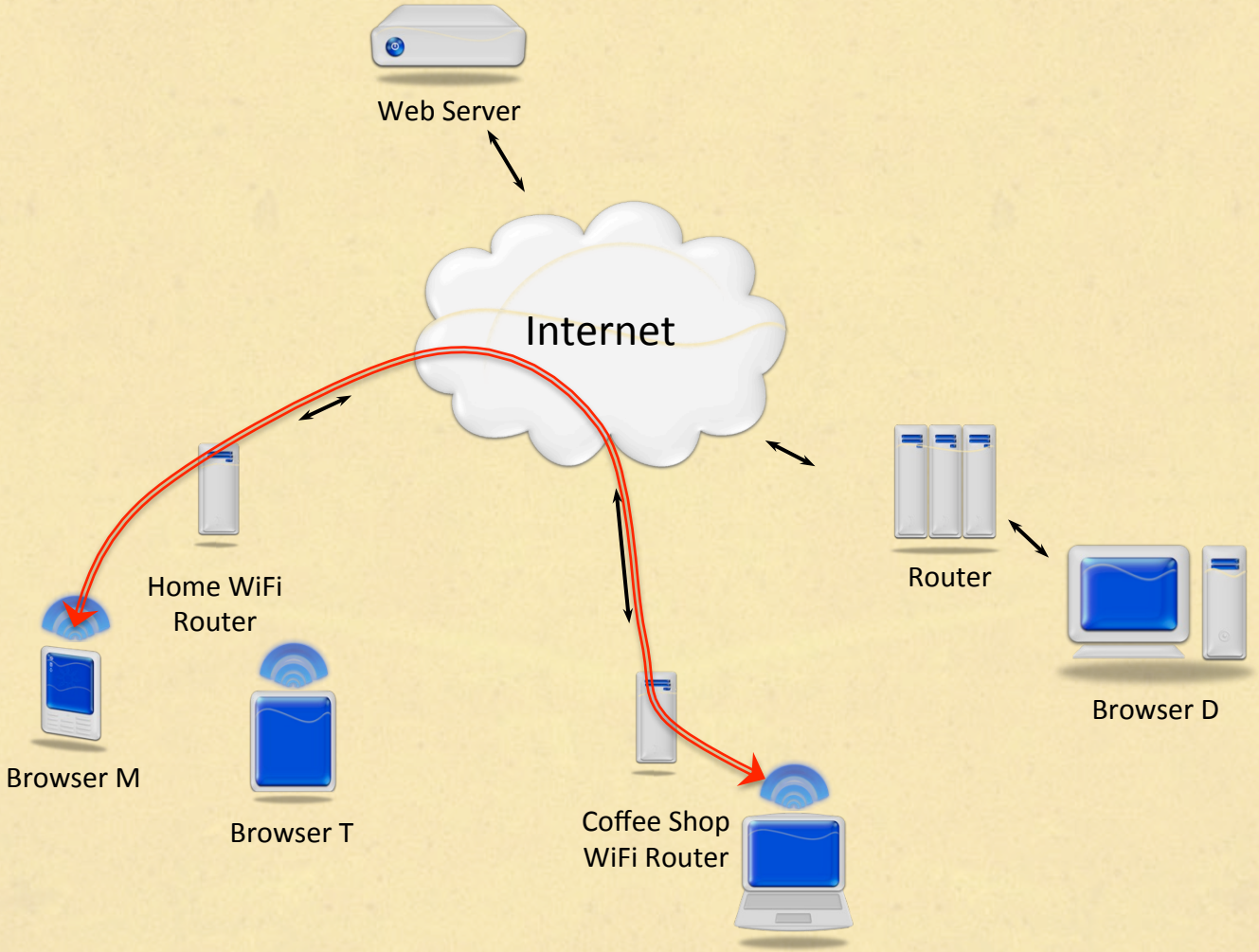
WebRTC Trapezoid



- Similar to SIP Trapezoid
- Web Servers communicate using SIP or Jingle or proprietary

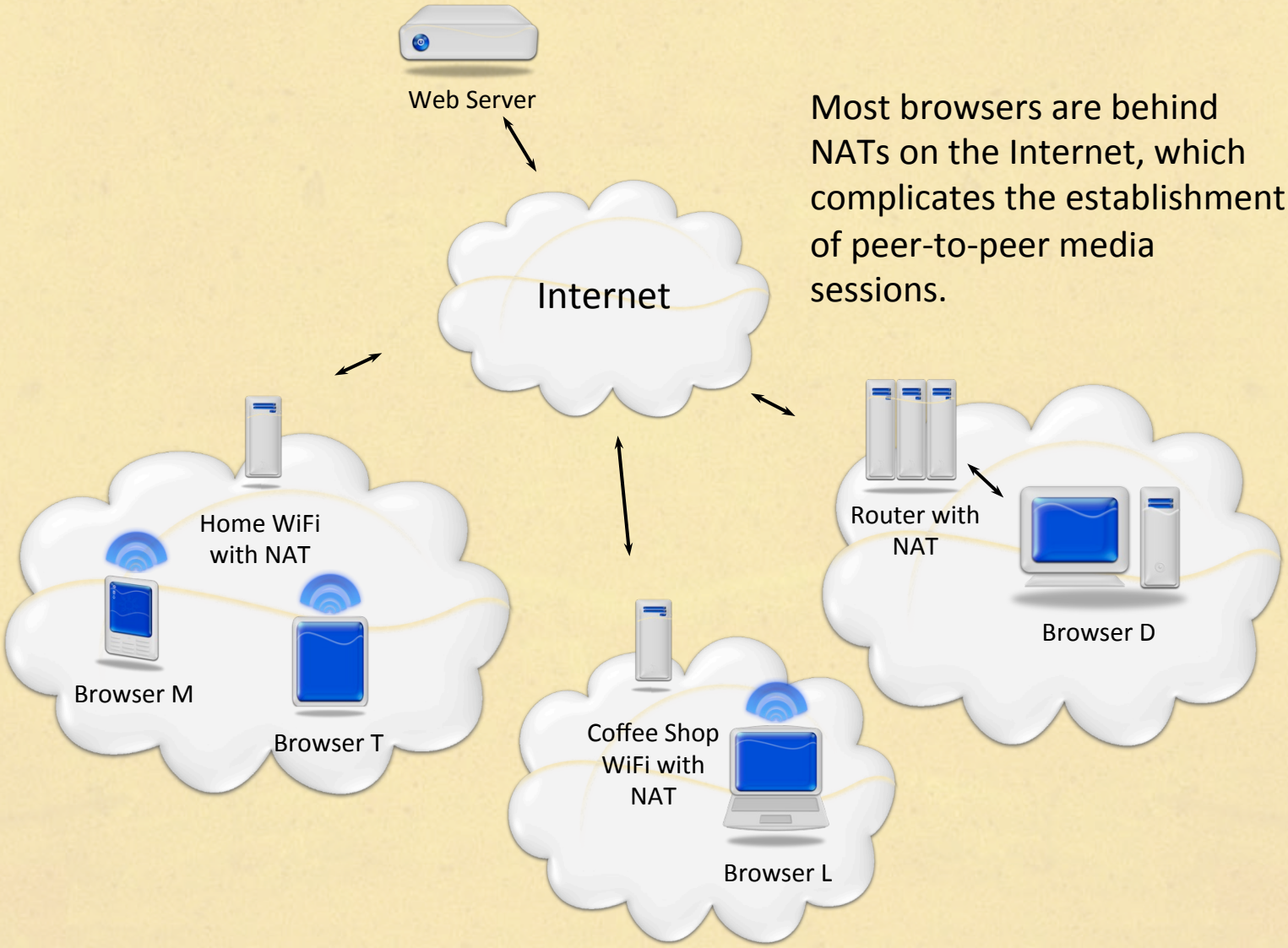
NAT Traversal and STUN

Peer-to-Peer Media with WebRTC



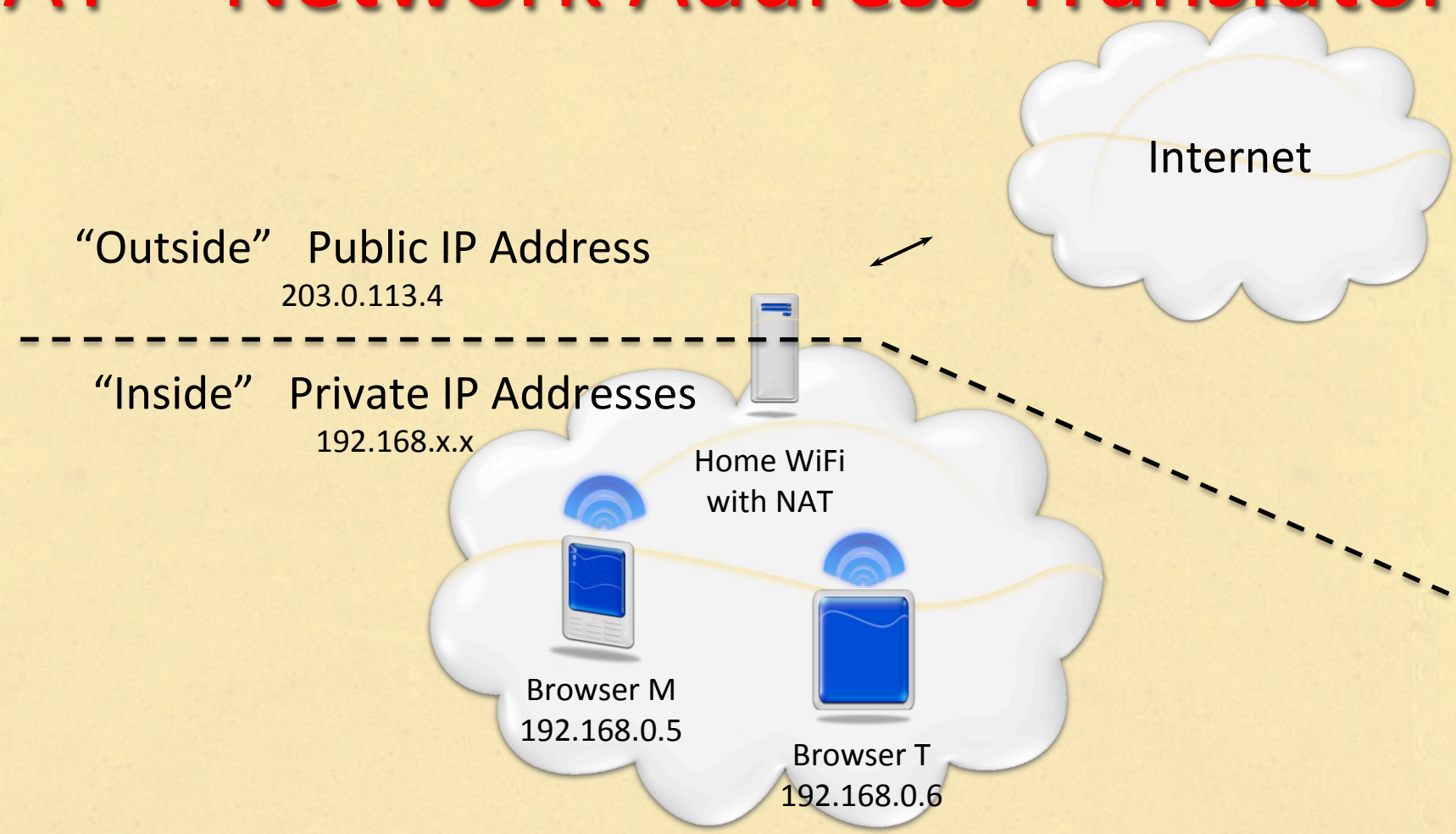
Media can go directly between browsers!

NAT Complicates Peer-to-Peer Media



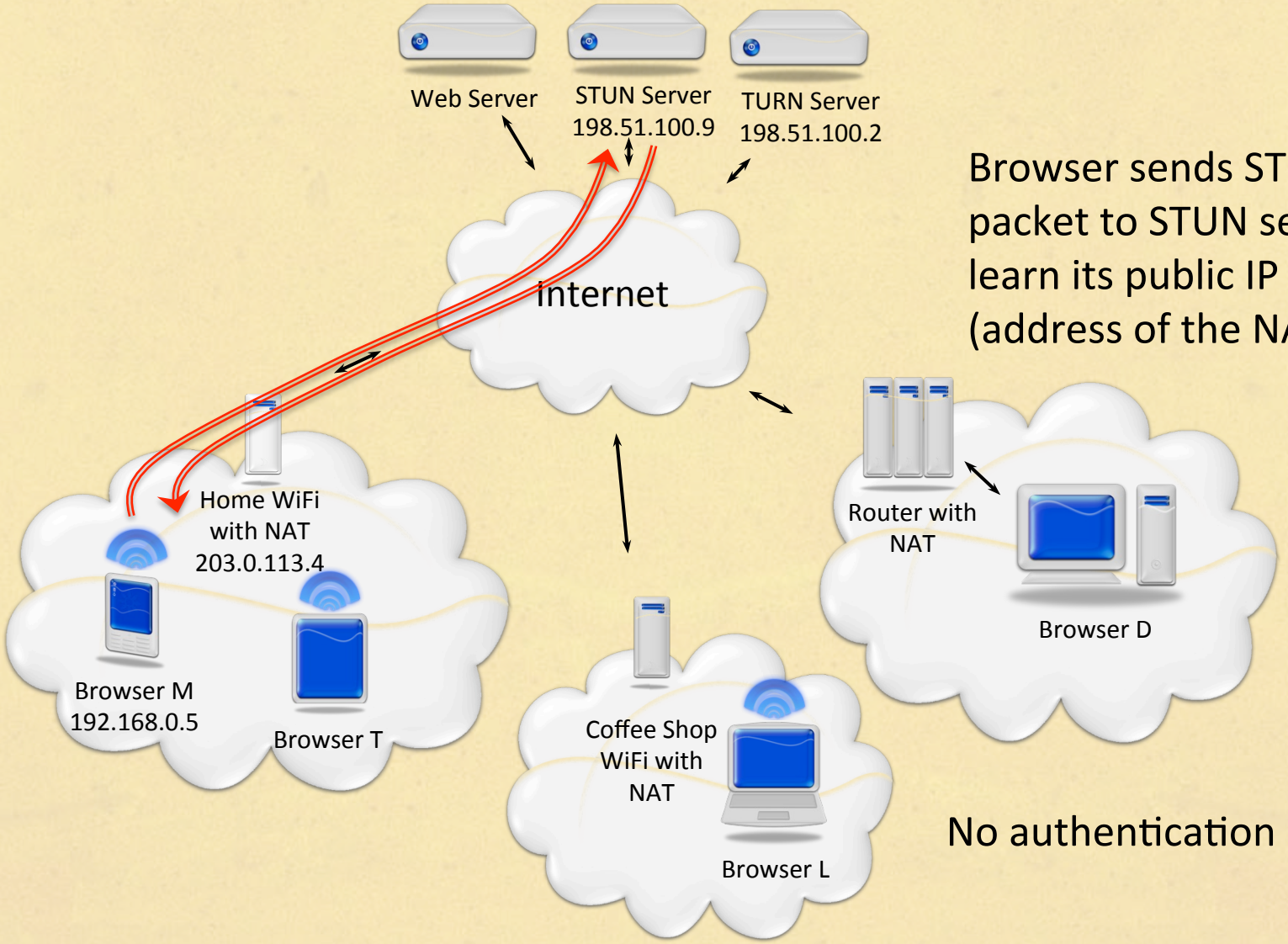
Most browsers are behind NATs on the Internet, which complicates the establishment of peer-to-peer media sessions.

NAT – Network Address Translator



A NAT maps an "inside" address to an "outside" address allowing multiple hosts to share the same IP address.

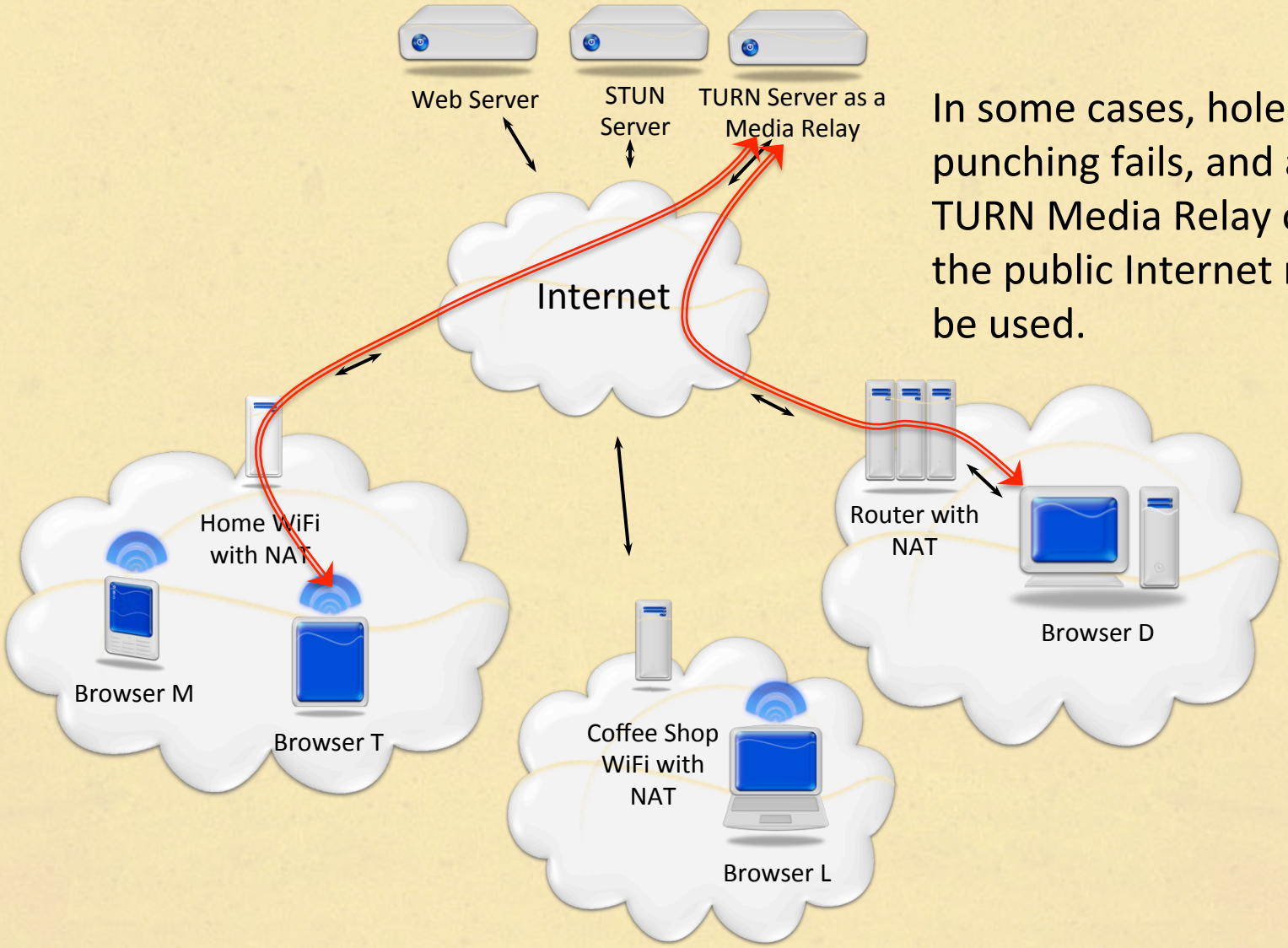
STUN Session Traversal Utilities for NAT



Browser sends STUN test packet to STUN server to learn its public IP address (address of the NAT).

No authentication of STUN!

TURN – Traversal Using Relay around NAT



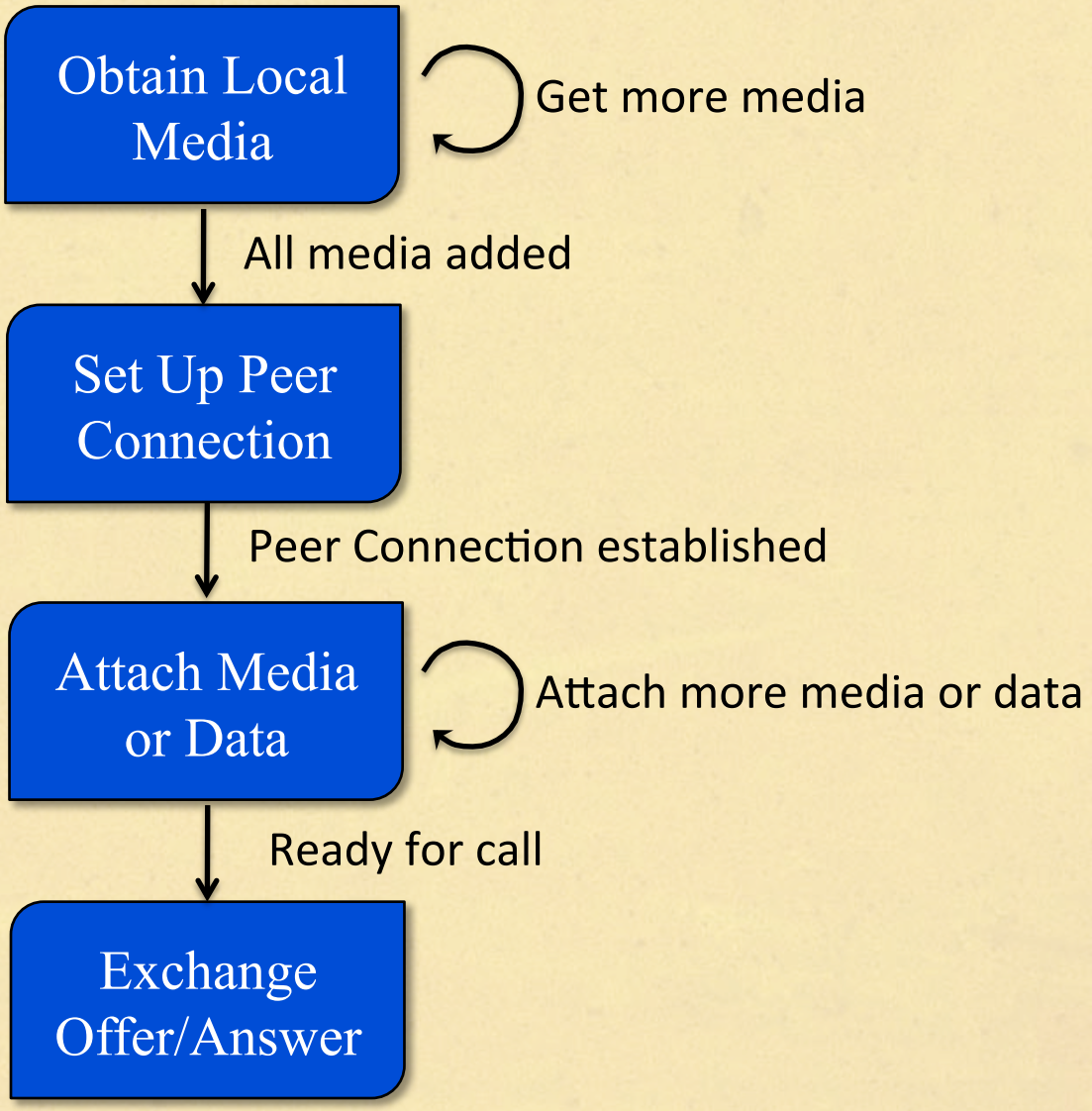
In some cases, hole punching fails, and a TURN Media Relay on the public Internet must be used.

What's New for Web Developers

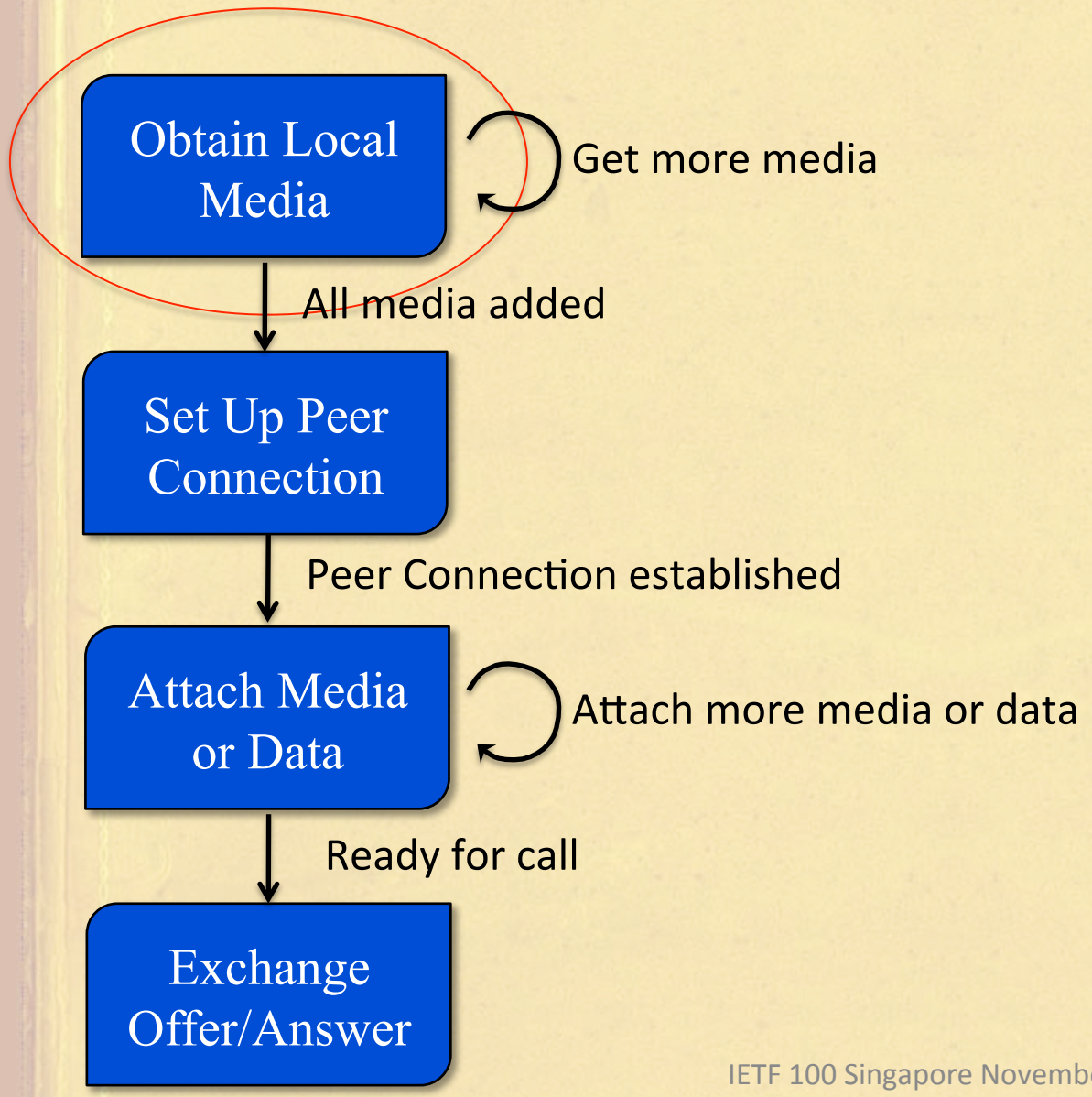
Two API pieces

- Local media capture
  - Camera, Microphone, <video> element, screen/ window
- Peer connection
  - Media and data between two client browsers (or arbitrary devices using native API)

WebRTC usage in brief

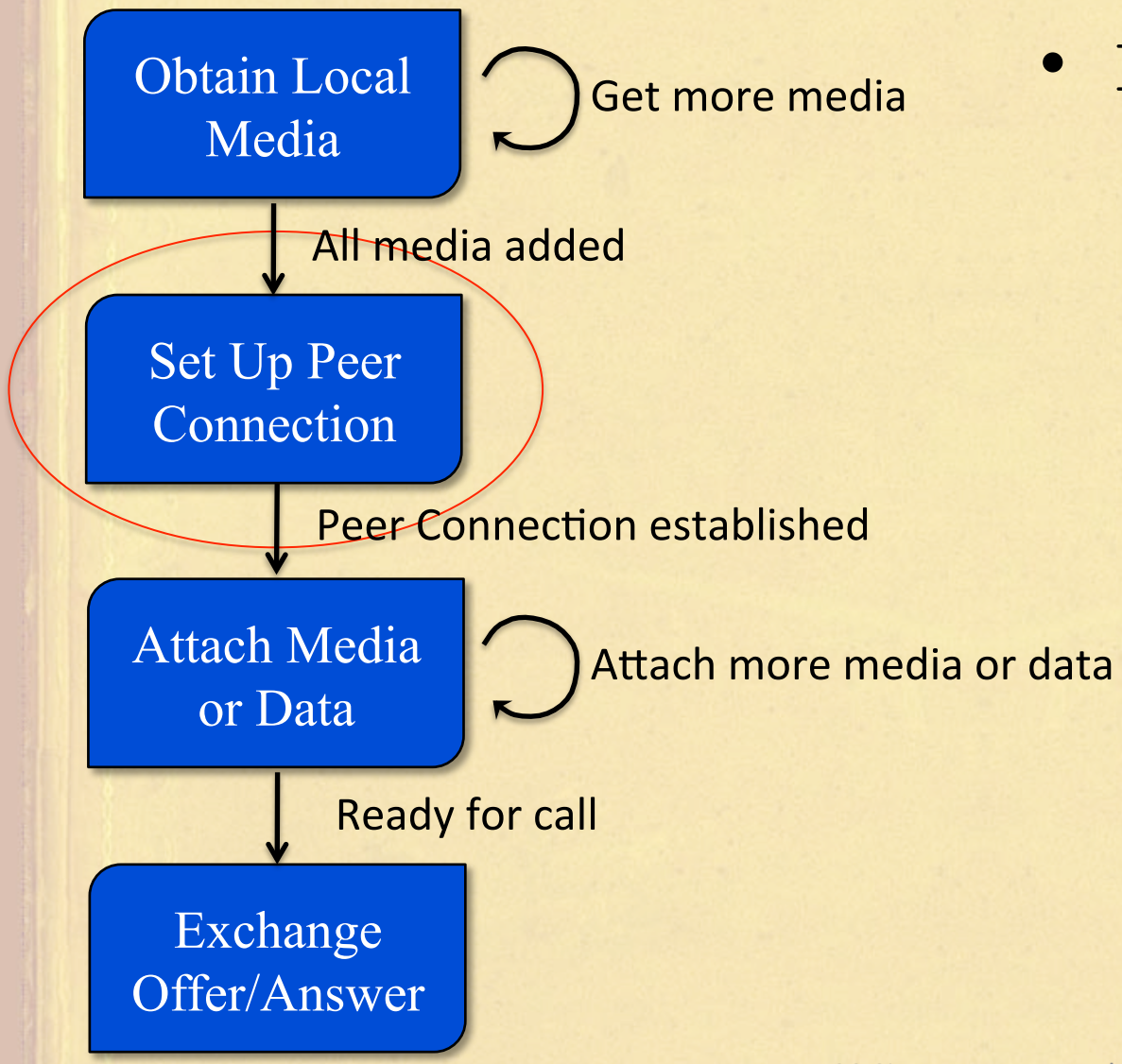


WebRTC usage in brief



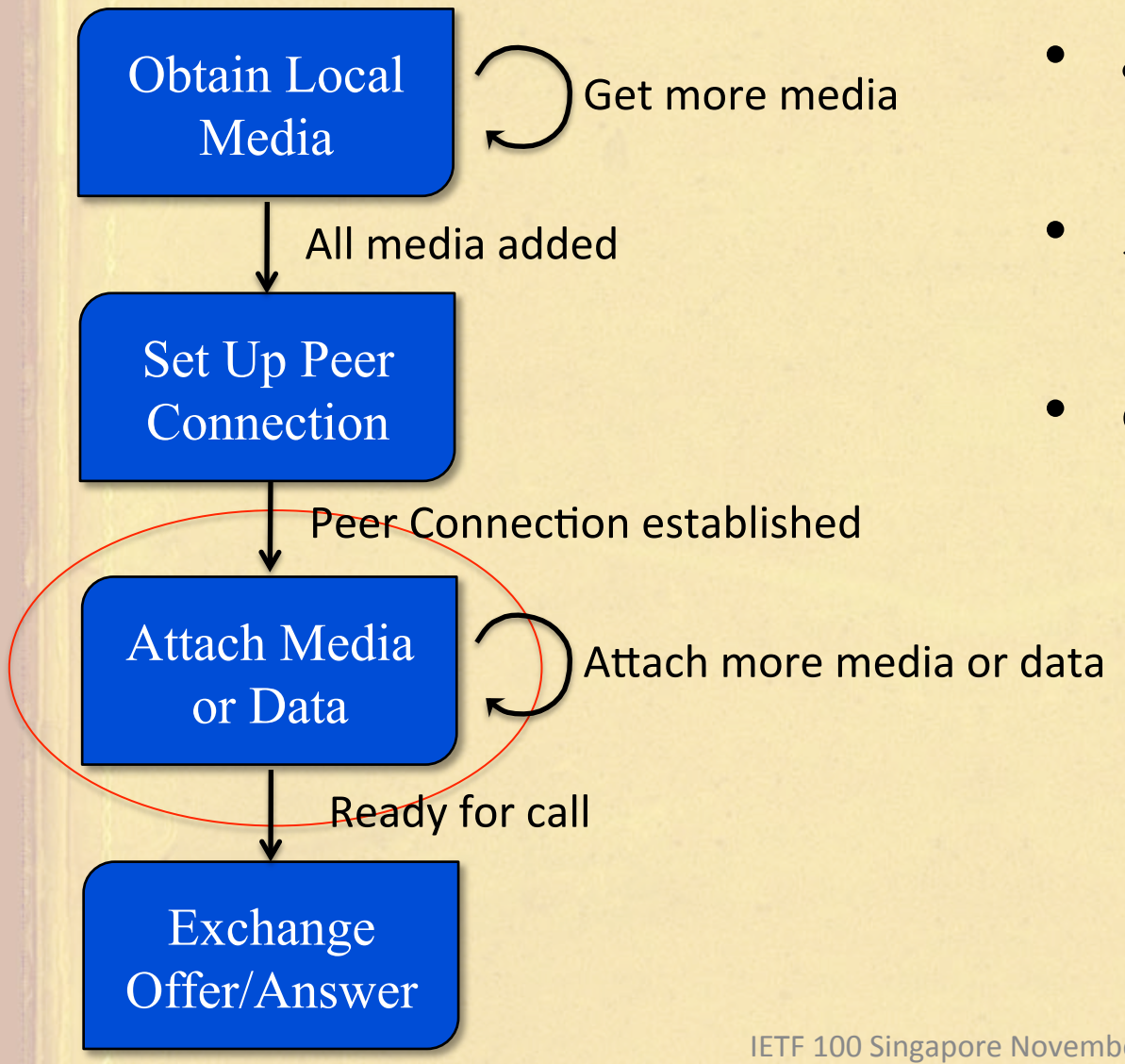
- `getUserMedia()`
  - Audio and/or video
  - Constraints
  - User permissions
    - Browser must ask before allowing a page to access microphone or camera
- `MediaStream`
- `MediaStreamTrack`

WebRTC usage in brief



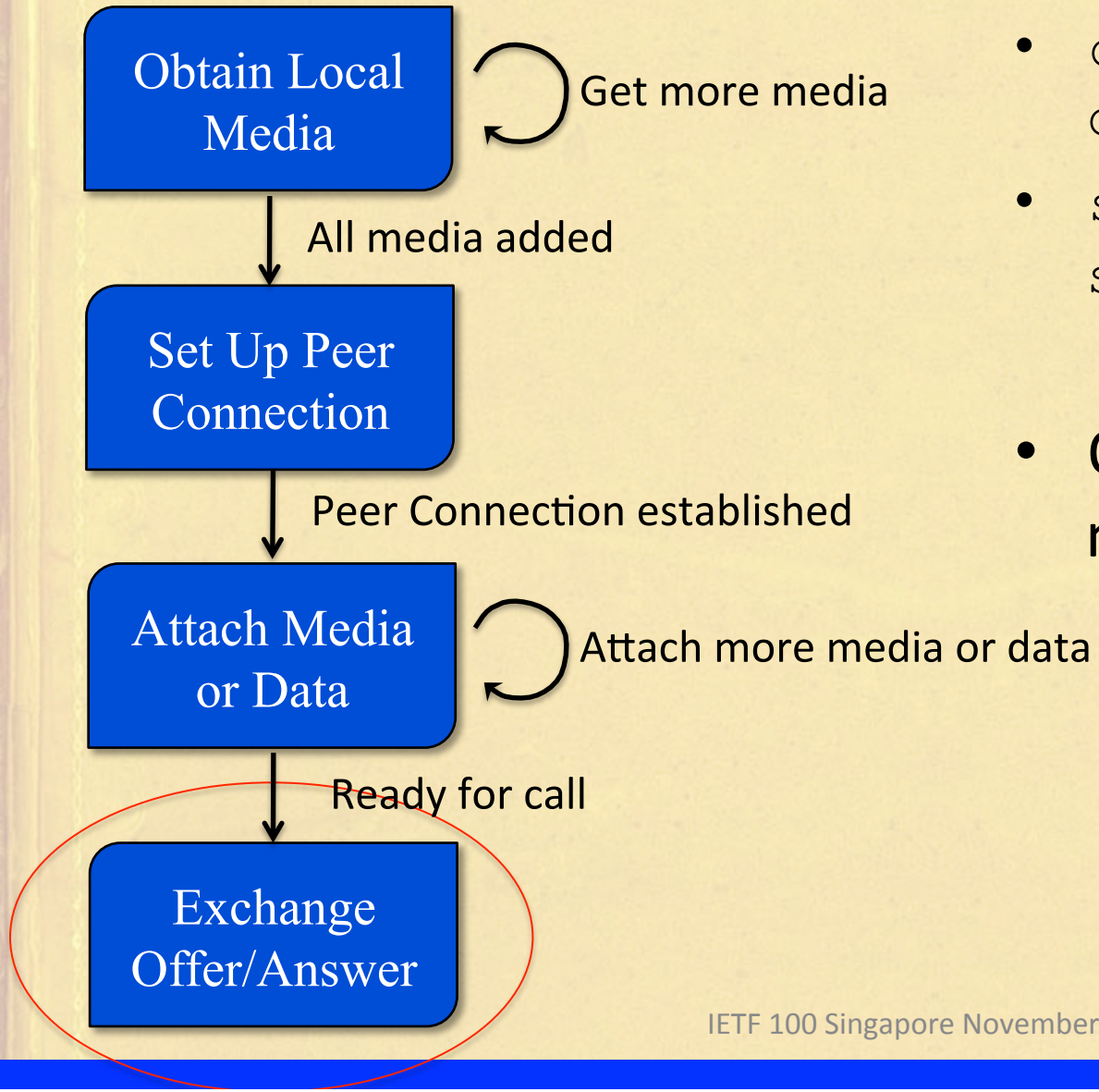
- `RTCPeerConnection`
  - Direct media
  - Between two peers
  - ICE processing
  - SDP processing
  - DTMF support
  - Data channels
  - Identity verification
  - Statistics reporting

WebRTC usage in brief



- `addTrack()`
  - Doesn't change media state!
- `removeTrack()`
  - Ditto!
- `createDataChannel()`
  - Depends on transport

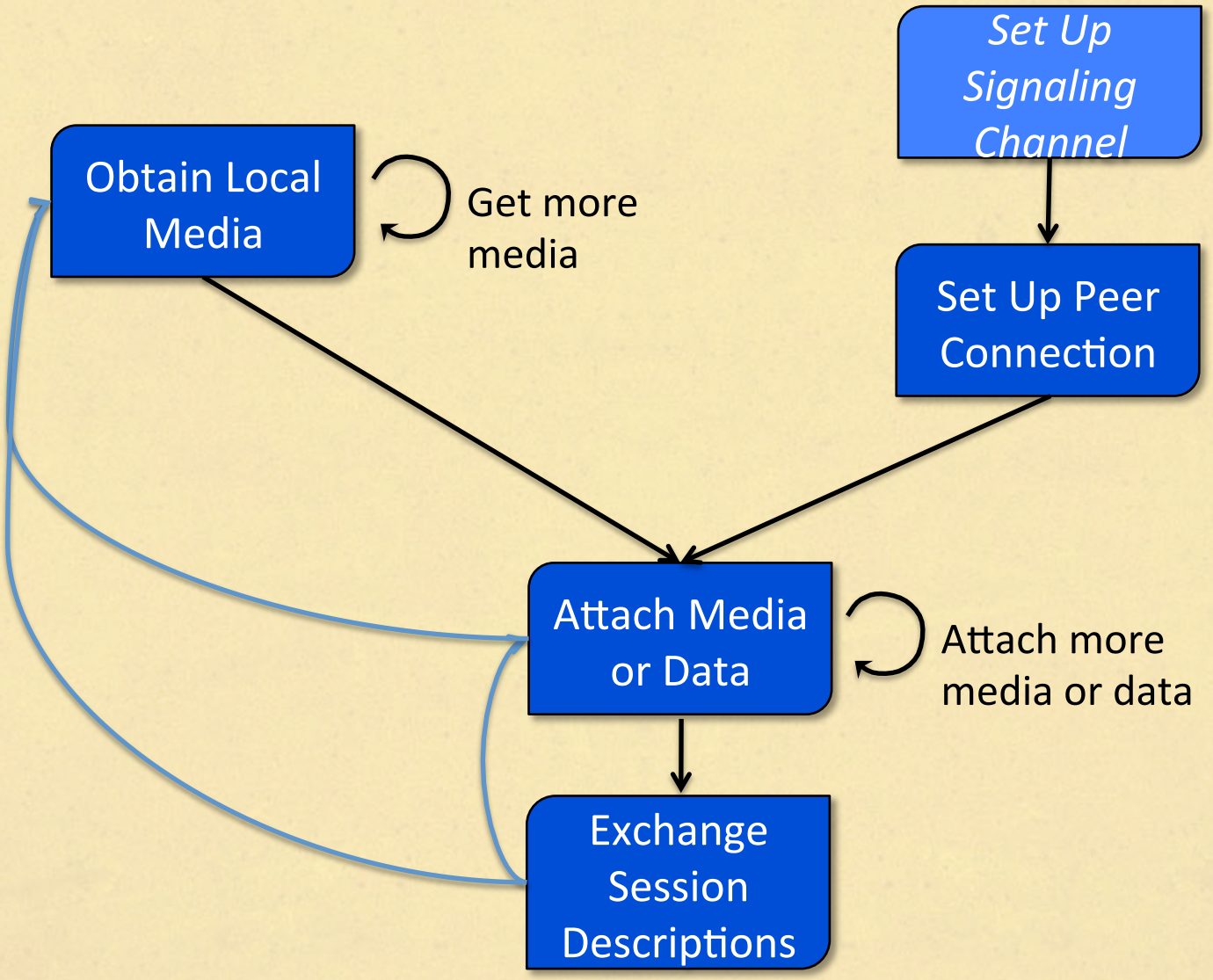
WebRTC usage in brief



- `createOffer()`, `createAnswer()`
- `setLocalDescription()`, `setRemoteDescription()`

- Offer/answer exchange needed for this to work

WebRTC usage in brief



Local Media

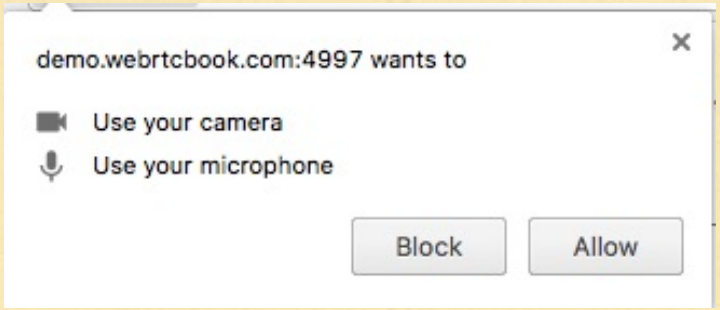
Media capture and use

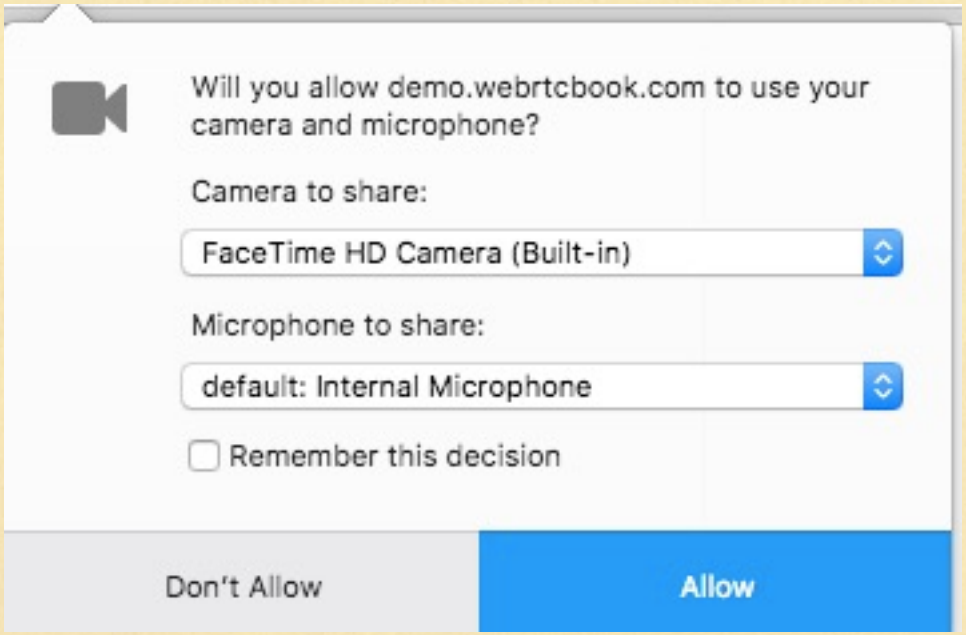
- **`navigator.mediaDevices.getUserMedia()`**
  - Request camera, microphone access
  - Permission check required for http page
- <video>**`.srcObject =`** <mediastream>
  - *Direct assignment*
  - Works for <audio> as well
- **`new MediaStream()`**
  - Can build from tracks in other streams

Browser Media Model: Sources and Sinks

- Sources can be:
  - static: files, RTSP, <canvas>, etc
  - dynamic:
    - local: webcam, microphone - need getUserMedia to access
    - remote: RTCPeerConnection or streaming media
- Sinks are <img>, <video>, and <audio> tags
  - They can be sized, which can cause scaling depending on the constraints on the source
- RTCPeerConnection can be both a source and a sink

Browser Prompts for Permission





Local: Tracks and Streams

- A **`MediaStreamTrack`**
  - is a handle to one real-time source of media of one type (audio/video/depth)
- A **`MediaStream`**
  - represents a collection of zero or more **`MediaStreamTrack`**s, of the same or different media types
  - indicates that the collected tracks are to be kept synchronized to the best ability of the browser

Peer Connections

RTCPeerConnection APIs

- Track stuff
  - addTrack(), removeTrack()
  - onaddtrack, onremovetrack
- Offer/answer stuff
  - createOffer(), createAnswer()
  - setLocalDescription(), setRemoteDescription()
- Much more

SDP Offer/Answer

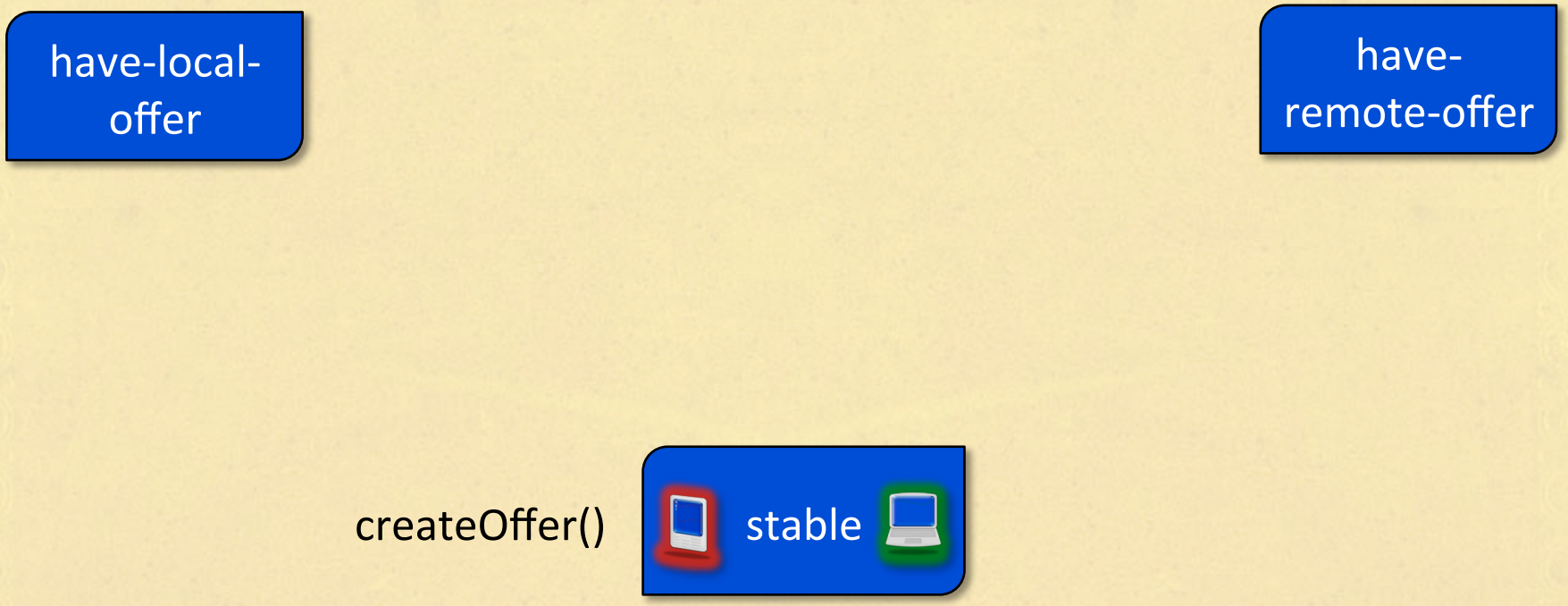
Session Description Protocol (SDP)

- Used by browser and WebRTC APIs to describe media session
  - RTP media flows, candidate transport addresses, codec information, media keying information
- SDP is widely used in SIP VoIP and video systems
- Browser use of SDP is based on Offer/Answer
  - Defined in JSEP - JavaScript Session Establishment Protocol
  - Negotiation proceeds according to Offer/Answer State Machine in next slides

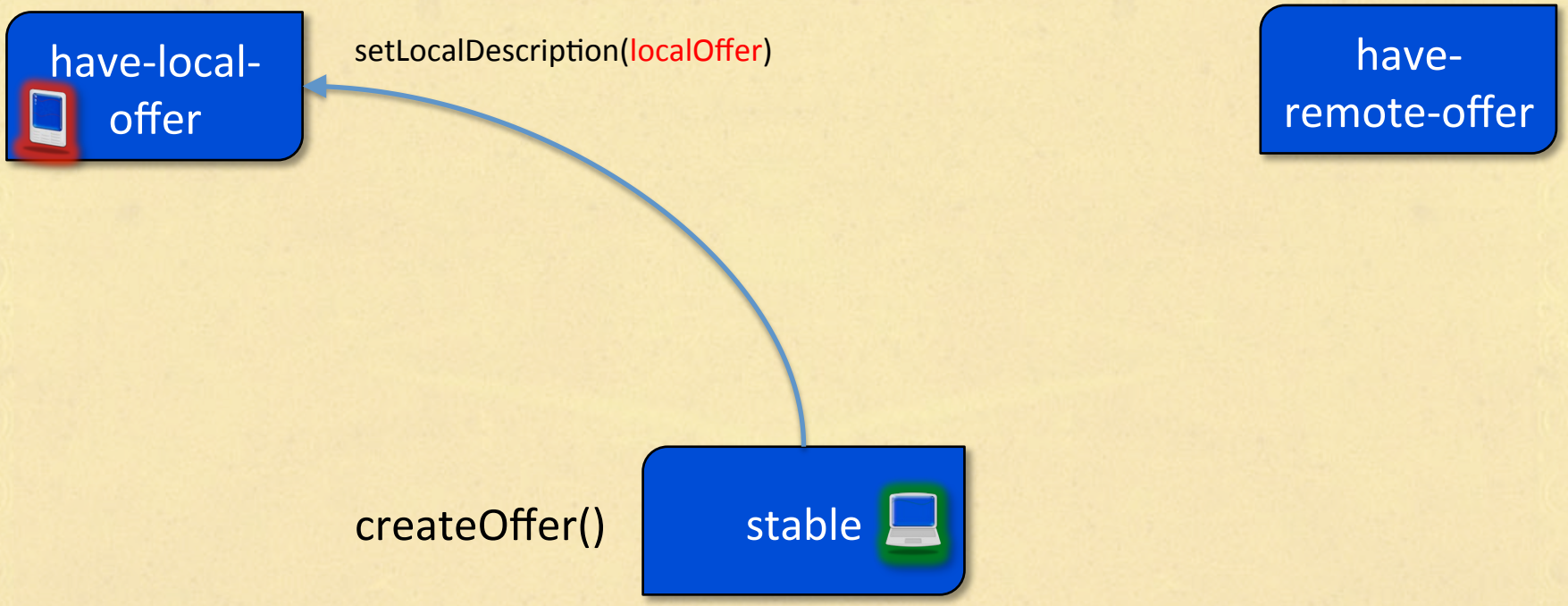
SDP Offer/Answer in RTCPeerConnection

- Peer Connection APIs treat the protocol-level config (SDP) as a blob
- **`createOffer(),createAnswer()`**
  - generate SDP
- **`setLocalDescription(), setRemoteDescription()`**
  - tell the browser which SDP to use

Offer/Answer State Machine

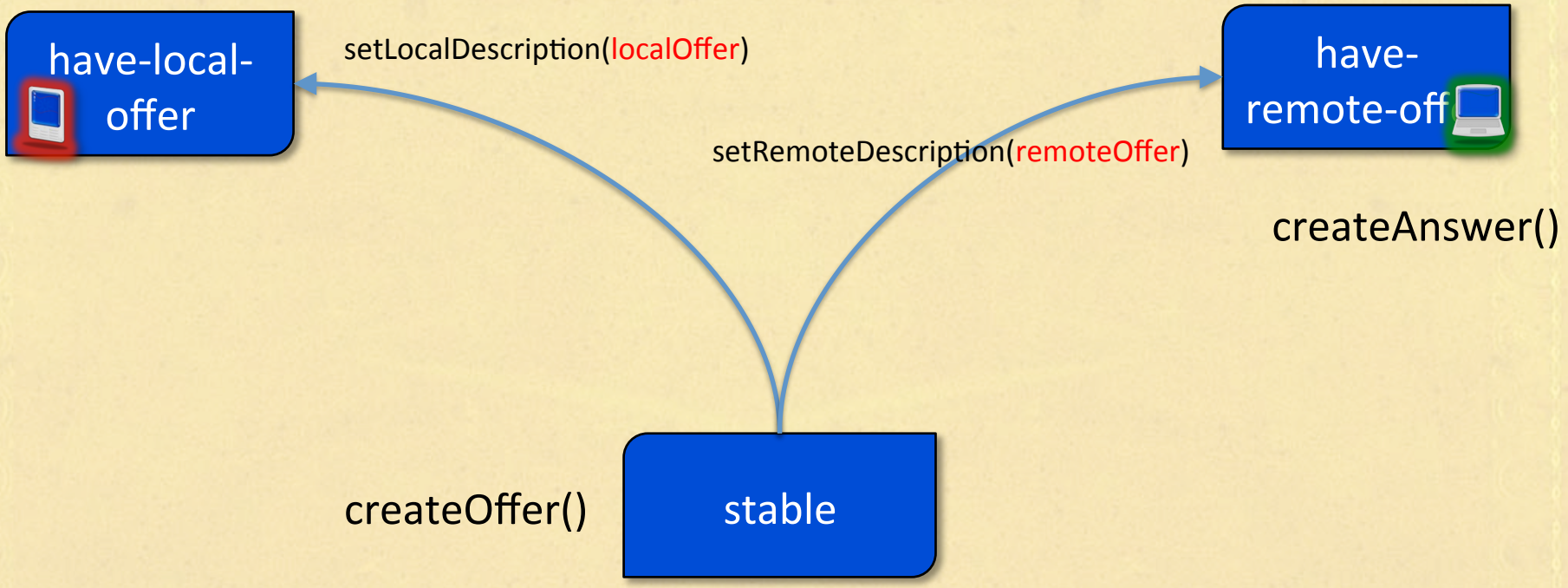


Offer/Answer State Machine



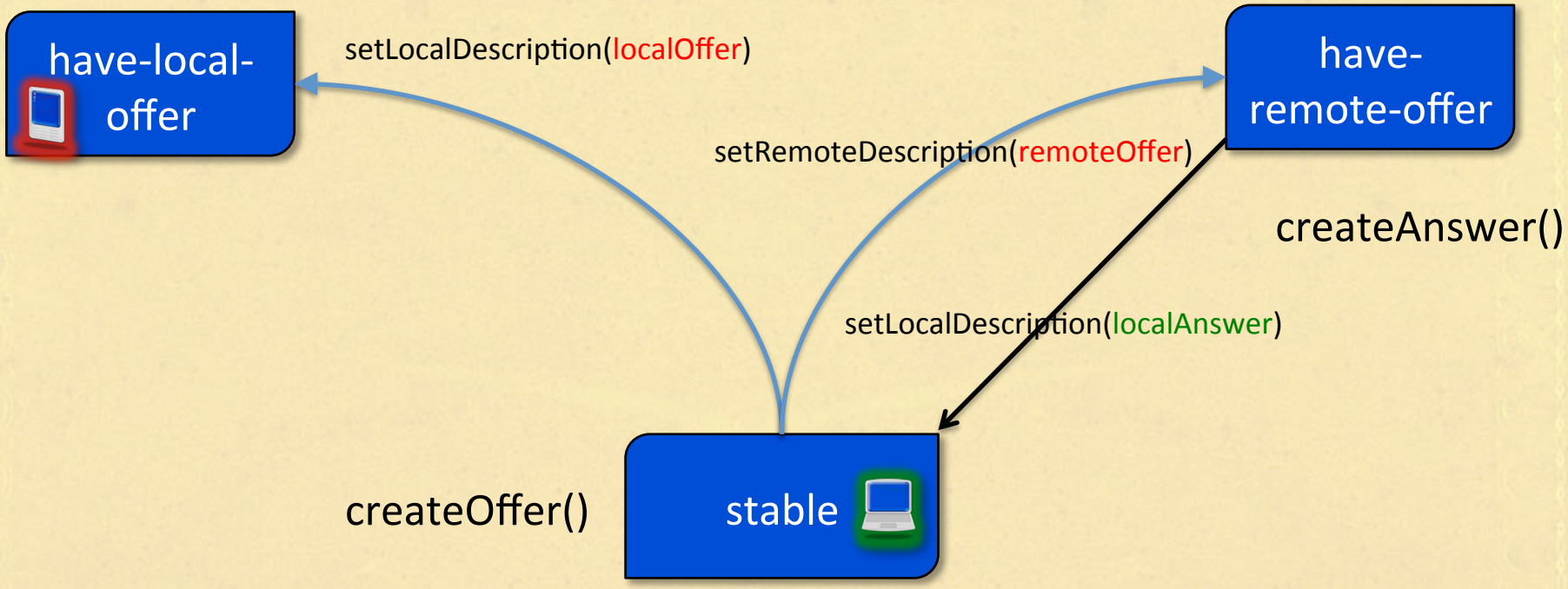
Offer/Answer State Machine

SDP offer sent over signaling channel to other browser



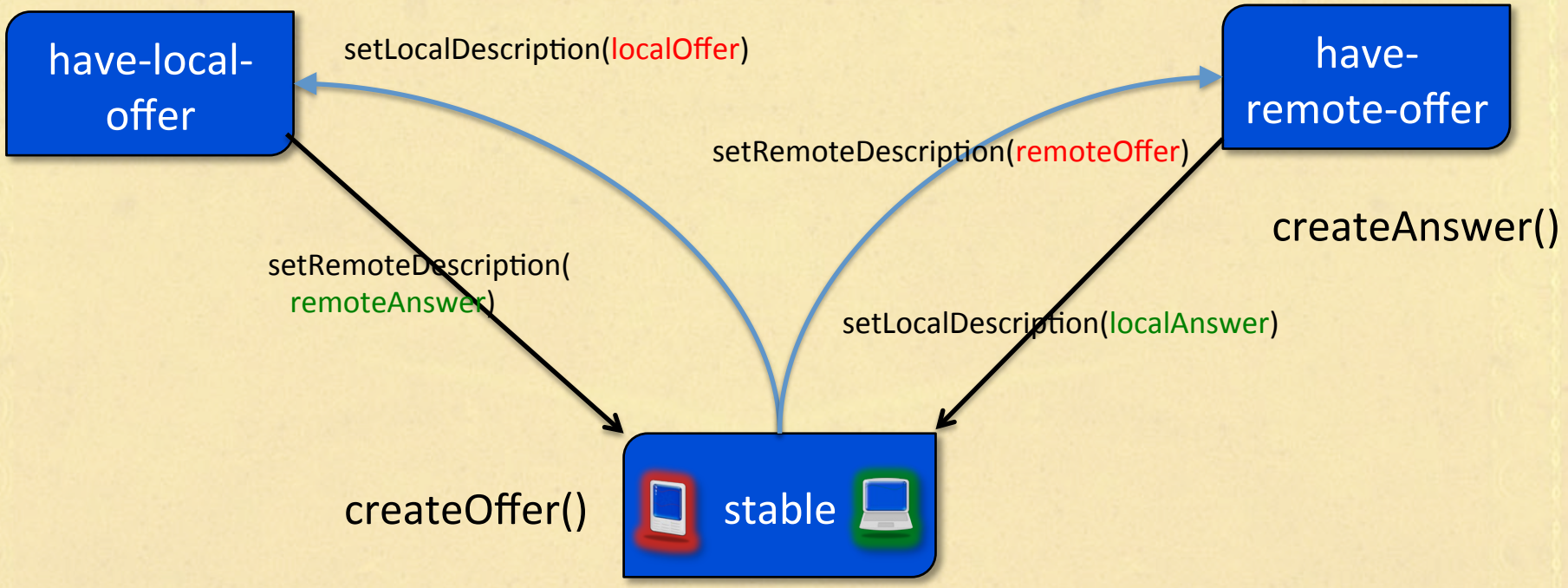
Offer/Answer State Machine

SDP offer sent over signaling channel to other browser



Offer/Answer State Machine

SDP offer sent over signaling channel to other browser



SDP answer sent over signaling channel to other browser

SDP Extensions in WebRTC

- A number of new SDP extensions developed for WebRTC. They include:
  - BUNDLE – a way to signal that a set of `m=` media lines (e.g. audio and video) should be multiplexed over the same transport address and port
    - `a=group:BUNDLE audio video`
  - MSID – a way to signal the Media Stream ID used in JavaScript in SDP
    - `a=msid`
  - Source-Specific Attributes – properties of a source
    - `a=ssrc`

SDP Offer/Answer Uses

Signaling

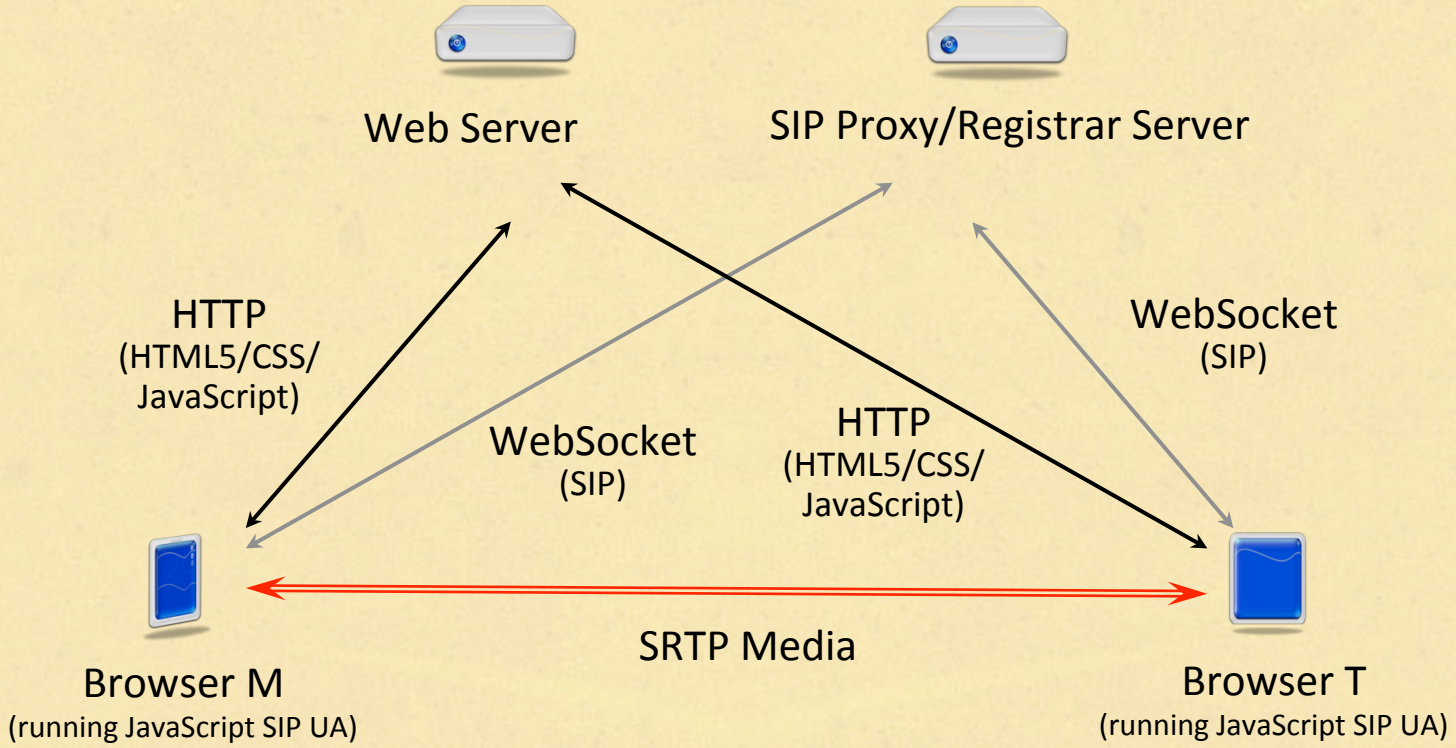
(not standardized)

WebRTC Signaling Approaches

- Signaling is required for exchange of candidate transport addresses and SDP blobs
- Many options – choice is up to web developer

Approach	Server Requirements	Advantages
WebSocket Proxy	WebSocket server with server code	No signaling infrastructure needed
XML HTTP Request	Web server with server code	No signaling infrastructure needed
SIP	SIP Registrar/Proxy Server which supports SIP WebSocket transport	Easy to interoperate with SIP endpoints or infrastructure, no server code needed
Jingle	XMPP server which supports XMPP WebSocket transport	Easy to interoperate with Jingle endpoints or infrastructure, no server code needed
Data Channel	WebSocket or web server to establish Data Channel	Low latency signaling and signaling privacy

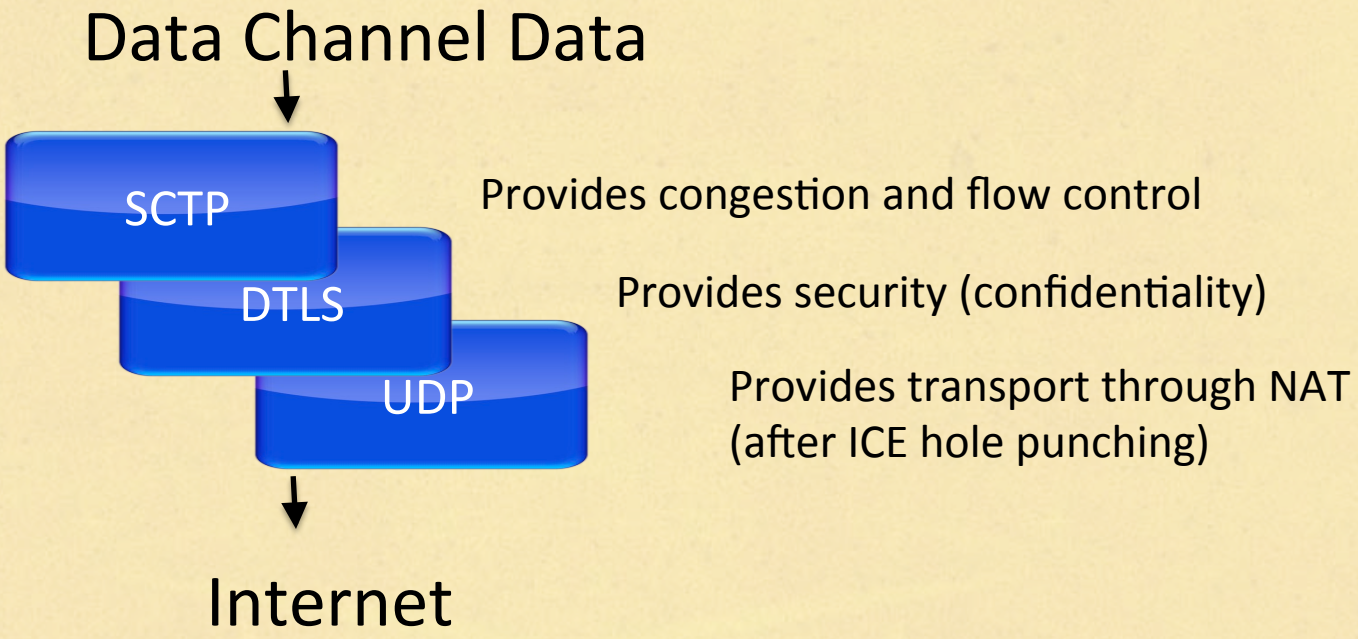
Example: WebRTC Signaling using SIP



- Browser runs a SIP User Agent by running JavaScript from Web Server
- SRTP media connection uses WebRTC APIs
- Details in RFC 7118 that defines SIP transport over WebSockets

Data Channel

Data Channel Protocols



- Data channel provides a non-media channel between browsers
- Stream Control Transport Protocol (SCTP) provides reliability, congestion control, and message delivery
- Multiplexed over same ports as RTP media

Data channel

- RTCPeerConnection.createDataChannel()
  - send() method for async message sending
  - onmessage handler for async message receipt
- Bidirectional by default
- Can send strings or ArrayBuffer types

Other details

Media Codecs

- Audio mandatory-to-implement:
  - Opus (RFC 6716): Narrowband to wideband Internet audio codec for speech and music
  - G.711 (RFC 3551): PCM audio encoding for PSTN interworking and backwards compatibility with VoIP systems
  - Telephone Events (RFC 4733): Transport of Dual Tone Multi Frequency (DTMF) tones
- Video mandatory-to-implement:
  - H.264 (RFC 6184): Common video codec (requires licensing)
  - VP8 (RFC 6386): Open source video codec

Security

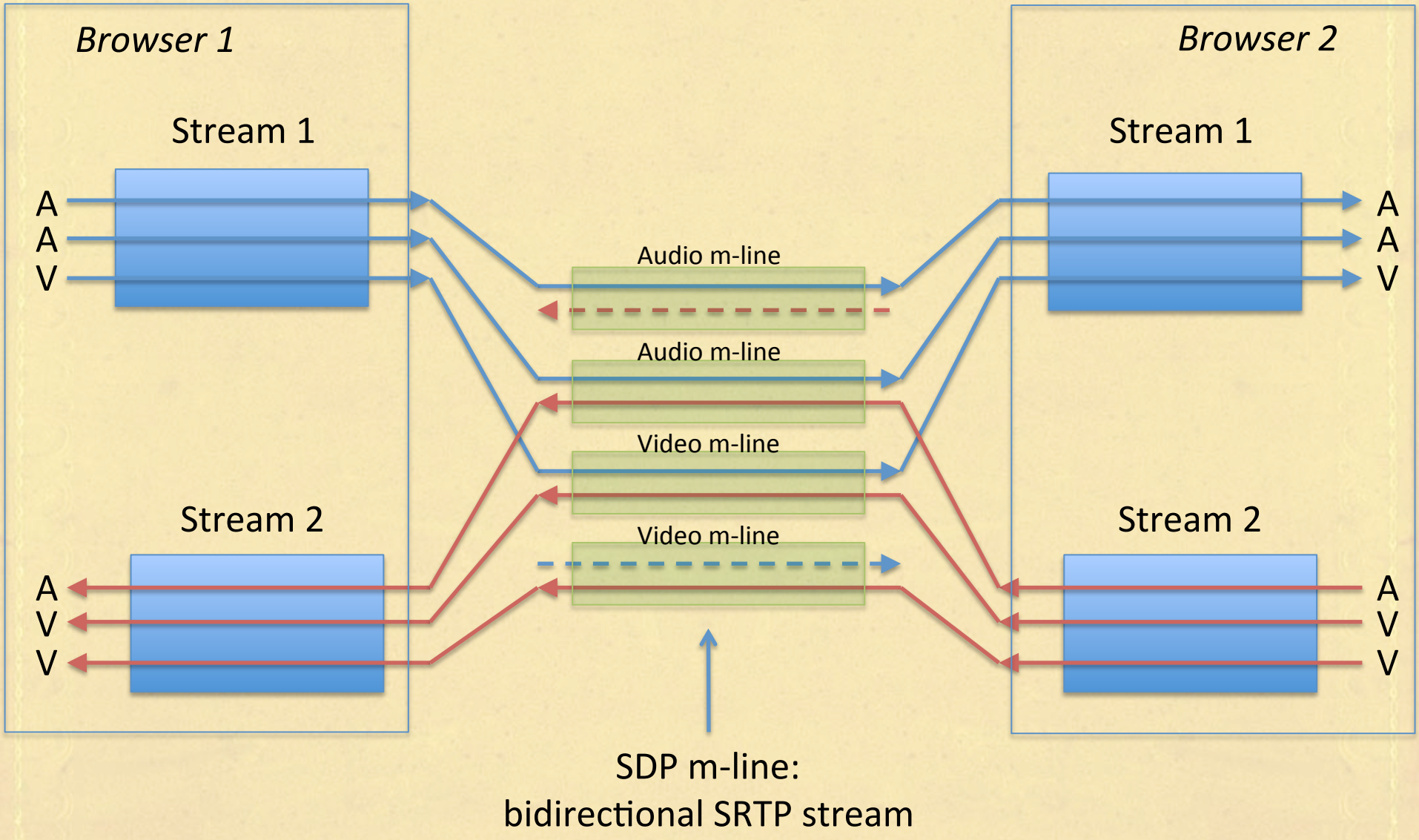
- Media is ***always*** encrypted in WebRTC
- Secure RTP (SRTP) is used to assure confidentiality and authentication of RTP and RTCP packets

Privacy

- SRTP, DTLS guarantee integrity of content but not the endpoint
- New Identity Proxy proposed in WebRTC
  - draft-ietf-rtcweb-security
  - Allows use of third-party identity service (e.g., Facebook Connect)
  - Browser signs SRTP keys using material from identity service, verified at other end using identity service

New Low-level Controls

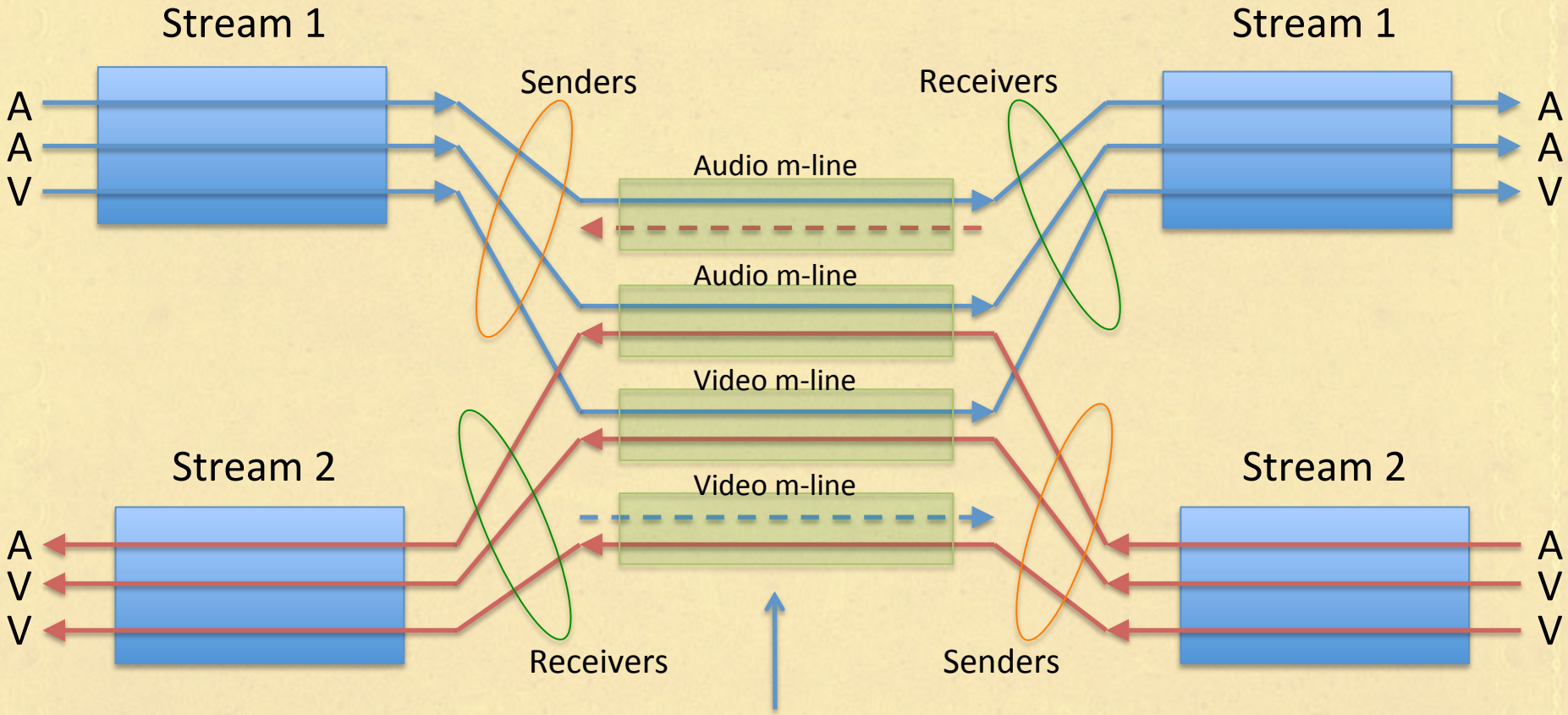
How it Really Works



Senders/Receivers

- WebRTC 1.0 recently added:
  - **`RTCRtpSender`**s – A handle to an outbound RTP stream (one-half of an m-line)
  - **`RTCRtpReceiver`**s – A handle to an inbound RTP stream (the other half of an m-line)
- For every track sent or received over a Peer Connection there is an associated sender or receiver. These give direct access to and control over the relevant RTP streams.

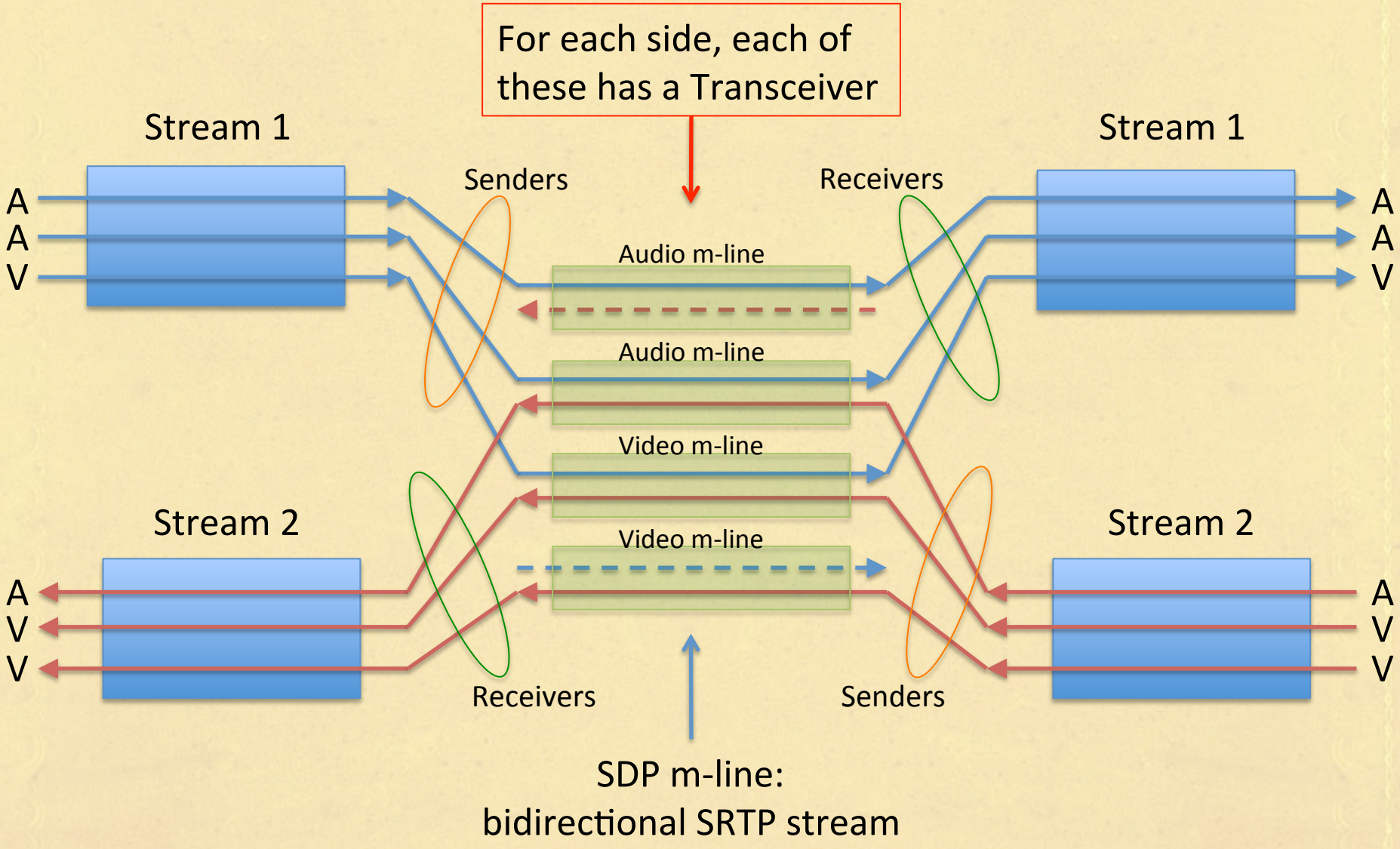
How it Really Works



Transceivers

- But that wasn't enough, because
  - at the SDP level every sender is paired with a receiver, even if only one has a track that is actively using it
- Enter the **`RTCRtpTransceiver`**. At each endpoint
  - There is precisely one transceiver for each m-line.
  - There is a sender and a receiver for each transceiver.
  - The mid of the m-line is the mid (media id) of the transceiver. It is unique per Peer Connection.

How it Really Works



Status of WebRTC APIs

- JavaScript APIs are being standardized by W3C
- Two main specifications
  - "WebRTC 1.0: Real-time Communication Between Browsers" (aka PeerConnection)
    - Candidate Recommendation: http://www.w3.org/TR/webrtc
    - Core is stable, just cleaning up edge cases now
  - "Media Capture and Streams" (aka getUserMedia)
    - Candidate Recommendation:
      - https://www.w3.org/TR/mediacapture-streams/
    - Oh so close . . .
- Need implementation experience at this point

Status of WebRTC Protocols

- Two references to check for this:
  - Primary is Cullen Jennings' WebRTC Dependencies draft draft-jennings-rtcweb-deps
  - RTCWEB documents can be followed as usual https://tools.ietf.org/wg/rtcweb/

Tools and Services

- To help WebRTC programming
  - Libraries, code snippets
    - EasyRTC, SimpleWebRTC.js, CoTurn (https://github.com/coturn/coturn)
  - SIP signaling
    - sipML5, JsSIP, reSIProcate
- To help WebRTC deployment
  - Hosted signaling services
    - PubNub, FireBase
  - Hosted STUN and TURN servers
    - XirSys, Twilio
  - Hosted SFU and/or network optimization
    - Jitsi, SwitchRTC, Agora.io

Higher-level APIs

- Alternate (higher-level) APIs
  - Twilio, TokBox
  - APIDaze
  - PeerJS, RTCMultiConnection

Great online resources

- Informational sites
  - http://webrtc.org
  - http://html5rocks.com/en/tutorials/webrtc/basics
  - http://webrtchacks.com
  - https://webrtcstandards.info
- Games/demos/apps
  - http://www.cubeslam.com
  - http://shinydemos.com/facekat
  - http://sharefest.me (github.com/Peer5/Sharefest)

What about deployments?

- Well-known services
  - Facebook Chat
  - Amazon Mayday (and now Chime)
  - Google Hangouts, Duo
- Platform embeddings
  - Every UC platform today
  - And most telecom providers
- Free(mium) comm services
  - Crowdcast.io
  - GoToMeeting Free (previously Hu.tt)
  - vLine
  - Talky.io
  - Appear.in
  - Gruveo
  - Room
  - Rabbit
  - GetARoom.io
  - UberConference