

# **JSEP Overview**

**Justin Uberti**  
**IETF 82.5**

# Topics

- Status
- Why JSEP?
- Theory of Operation
- Example Call Setup
- Implementation Considerations
- Contrast between JSEP and ROAP

# Status

- draft-uberti-rtcweb-jsep-00 posted last week
- -01 about to be posted; fixes several issues
  - Clarified what operations change state
  - Fixes asymmetry in original offer/subsequent offer call ordering
  - Associates candidates with proper m= lines
  - Knows the number of m= lines to gather for
  - Tells app when candidate gathering is complete, and provides getters for full lists of candidates
  - connect() renamed to startIce()
  - SDP\_PRANSWER added to allow non-final answers

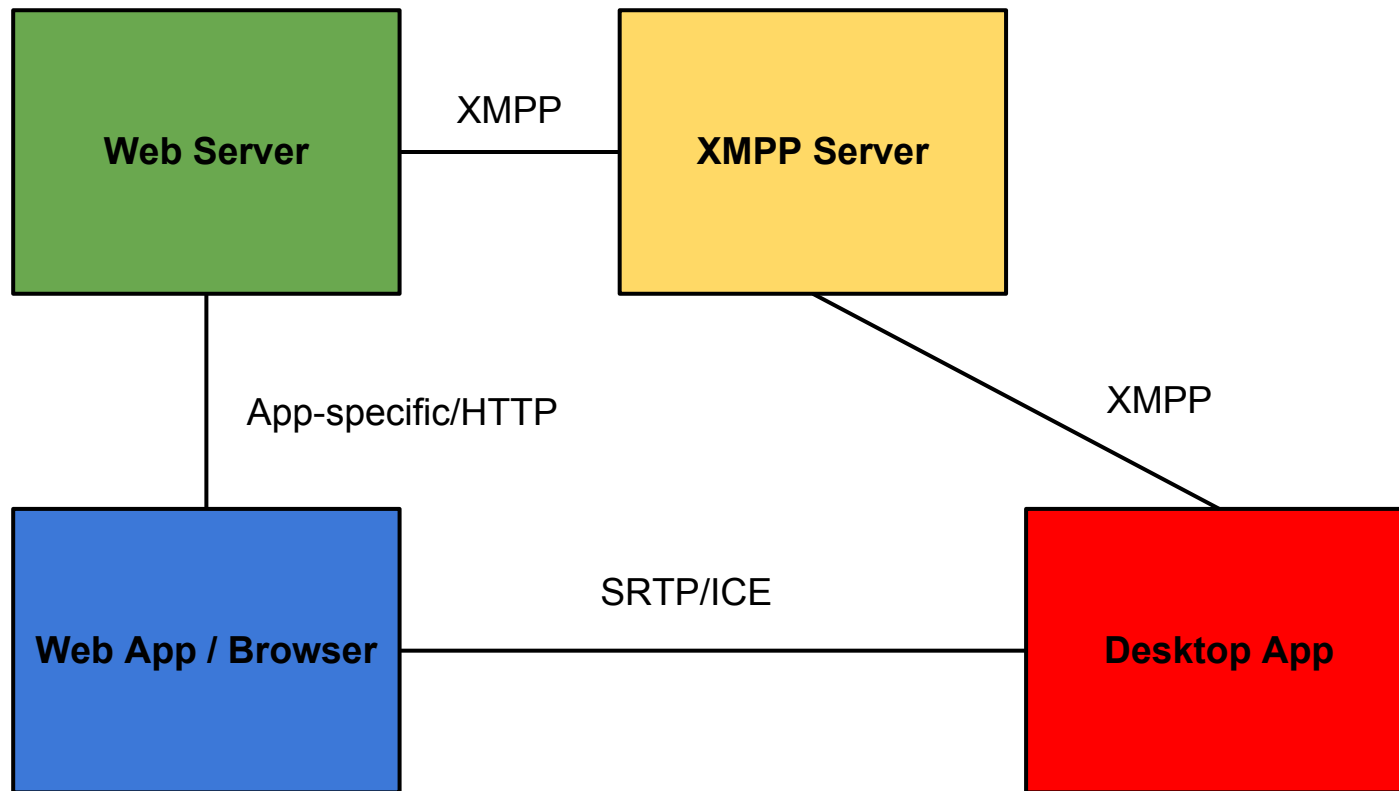
# Goals

- Allow easy translation to common signaling protocols and architectures
- Support early transport negotiation
- Allow local description to be changed by app
- Change session parameters at any time
- Allow direct manipulation of session state
- Give app as much flexibility as possible, now and in the future

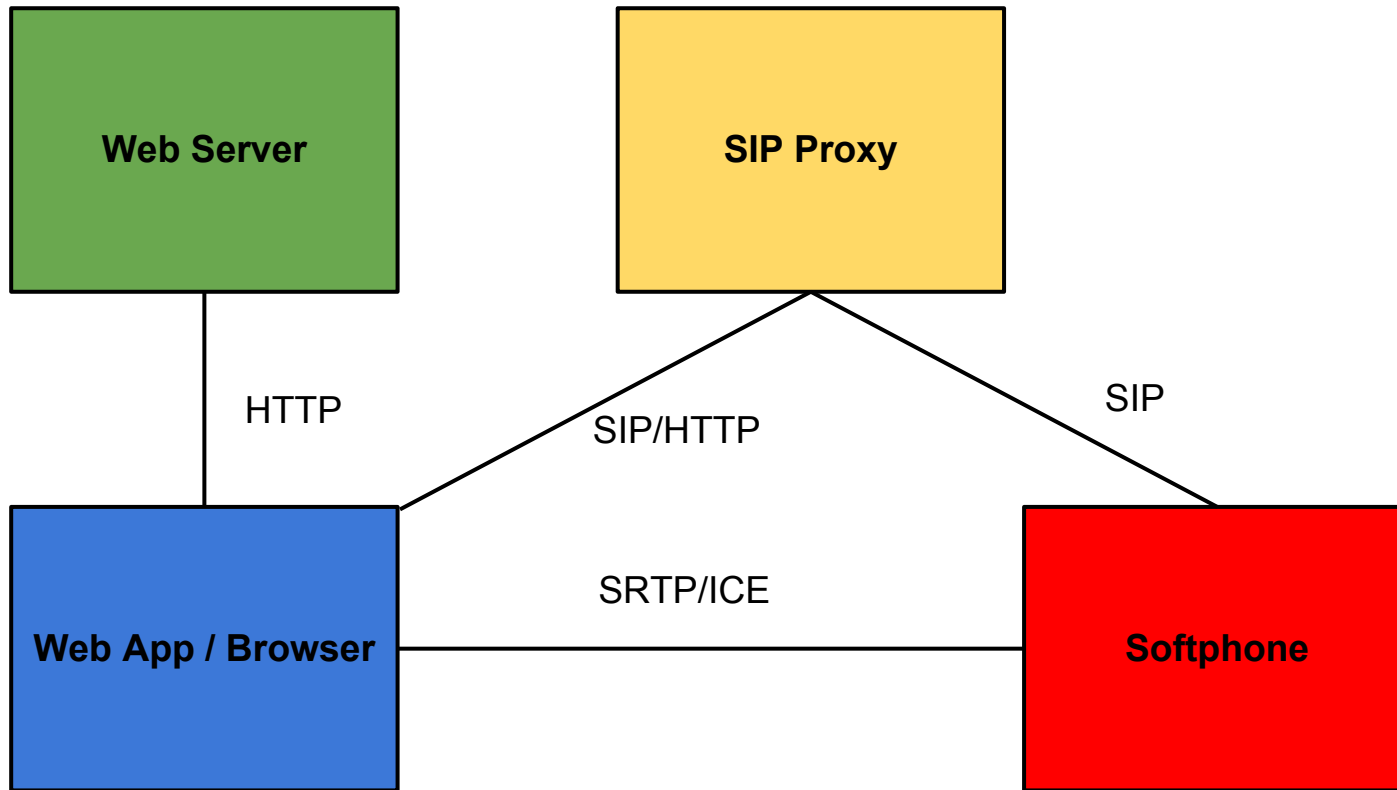
# Non-goals

- Super-simple API
- Replace SDP
- Offer generation in JS (at least not right now)

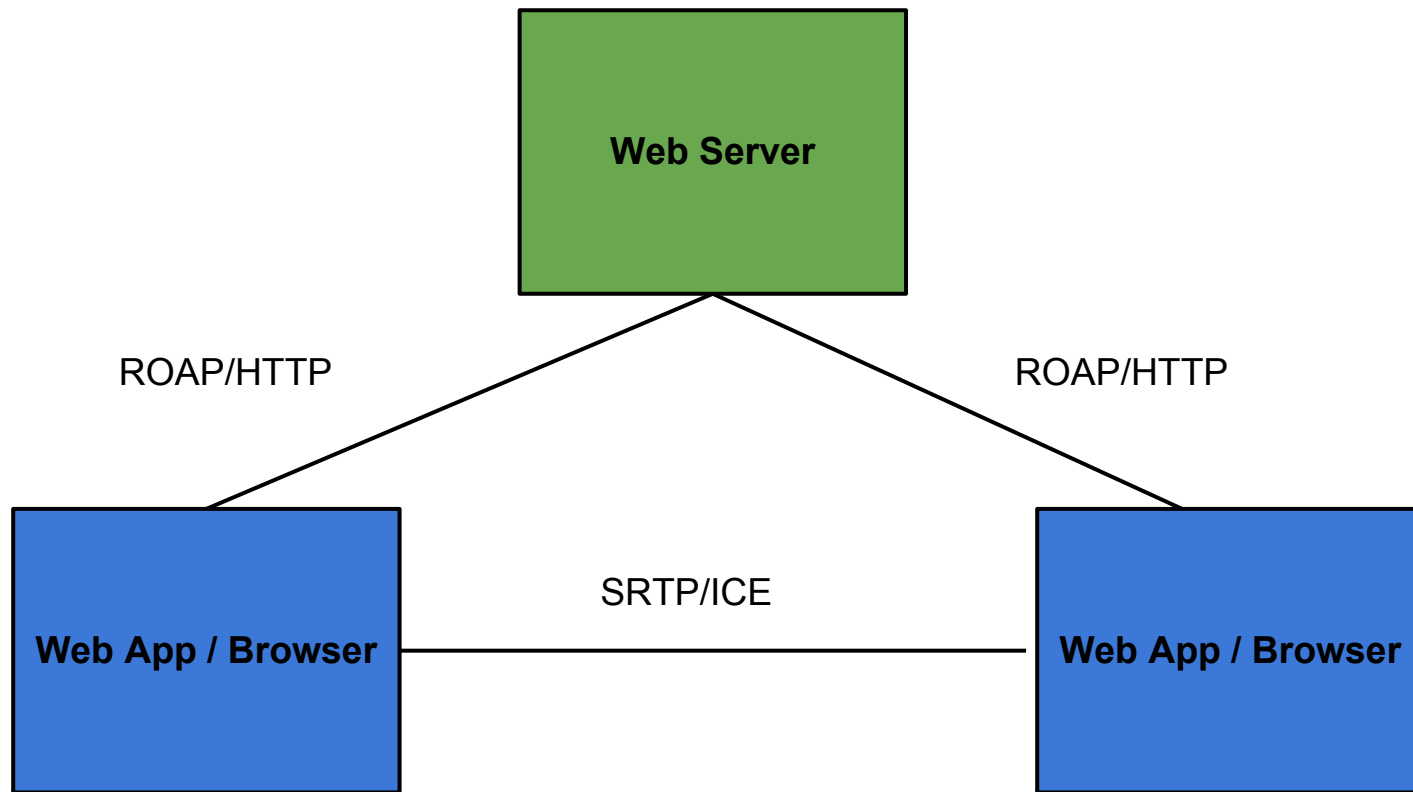
# Web App to Legacy Client



# IP SIP In Browser

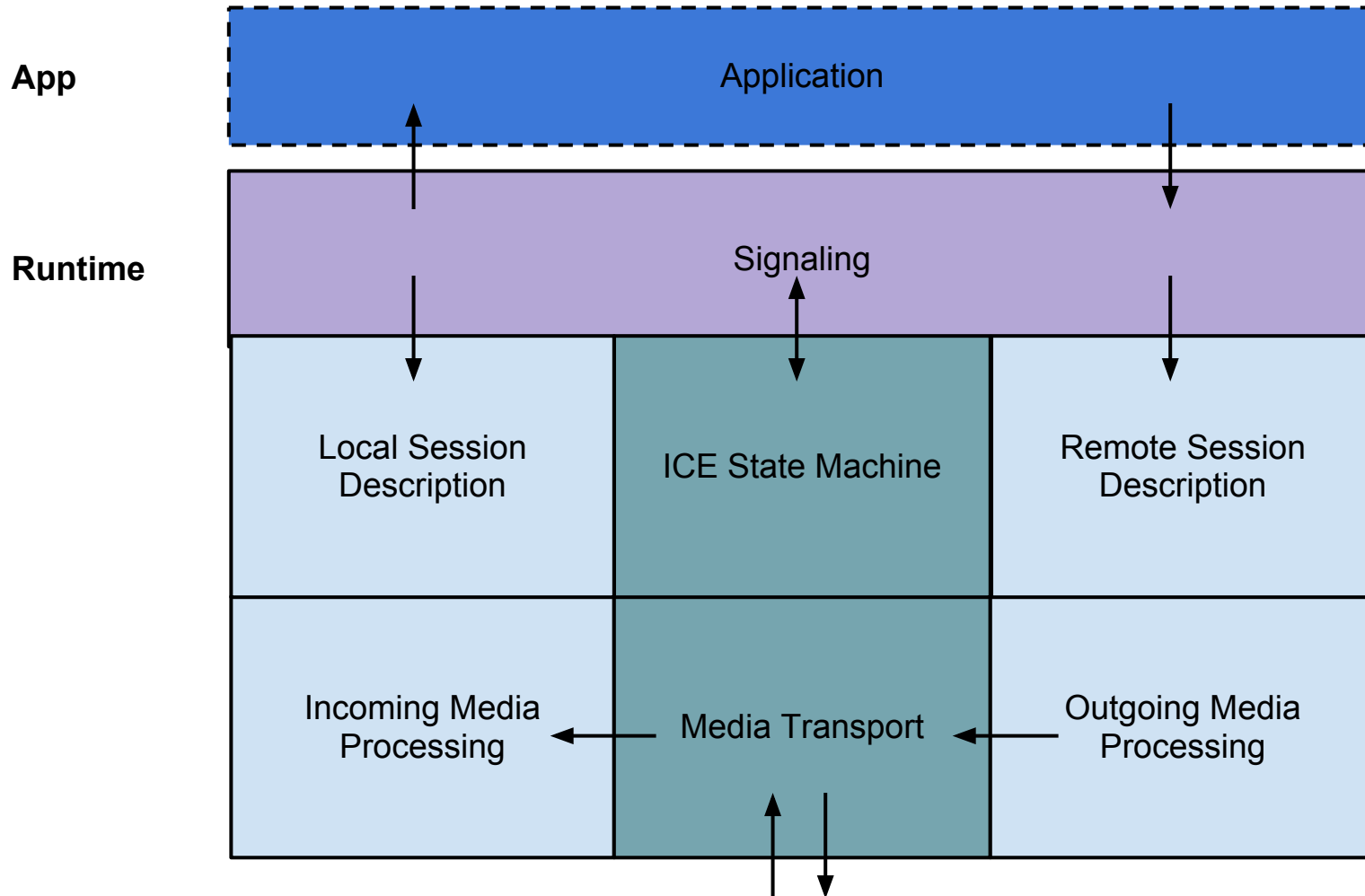


# Web to Web





# Conventional App Diagram



# Differences in Signaling

**Different signaling protocols have different features and mechanisms**

- Candidate handling
- Glare handling/Tie breaking
- Adding or removing sources or sessions
- RTP Session updates

# Core Problem

**It's hard to have a generic signaling mechanism that can map faithfully to all signaling state mechanisms**

# Needs of the Media Layer

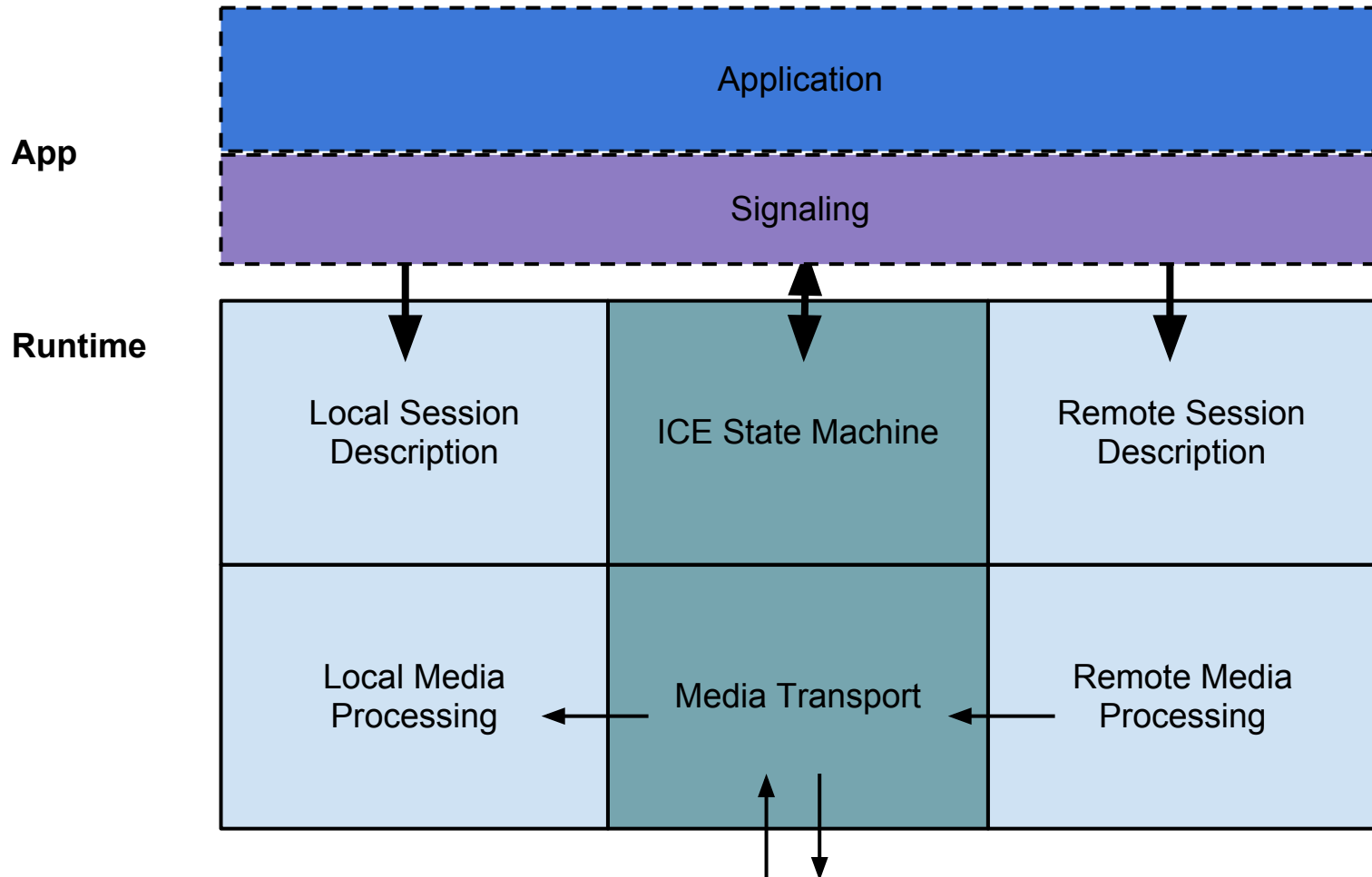
- Local format description
  - What I want/am going to do
- Remote format description
  - What the other side wants/is going to do
- Local/remote transport info
  - Where is media going to go, and how

**Signaling is a mechanism to obtain this information**

# JSEP Key Concepts

- Signaling and transport are separated
- Signaling state moved into application code
- Media controlled via local and remote session descriptions (SDP blobs)
- **The "how" of the signaling is left to the application; only the results of the signaling matter**

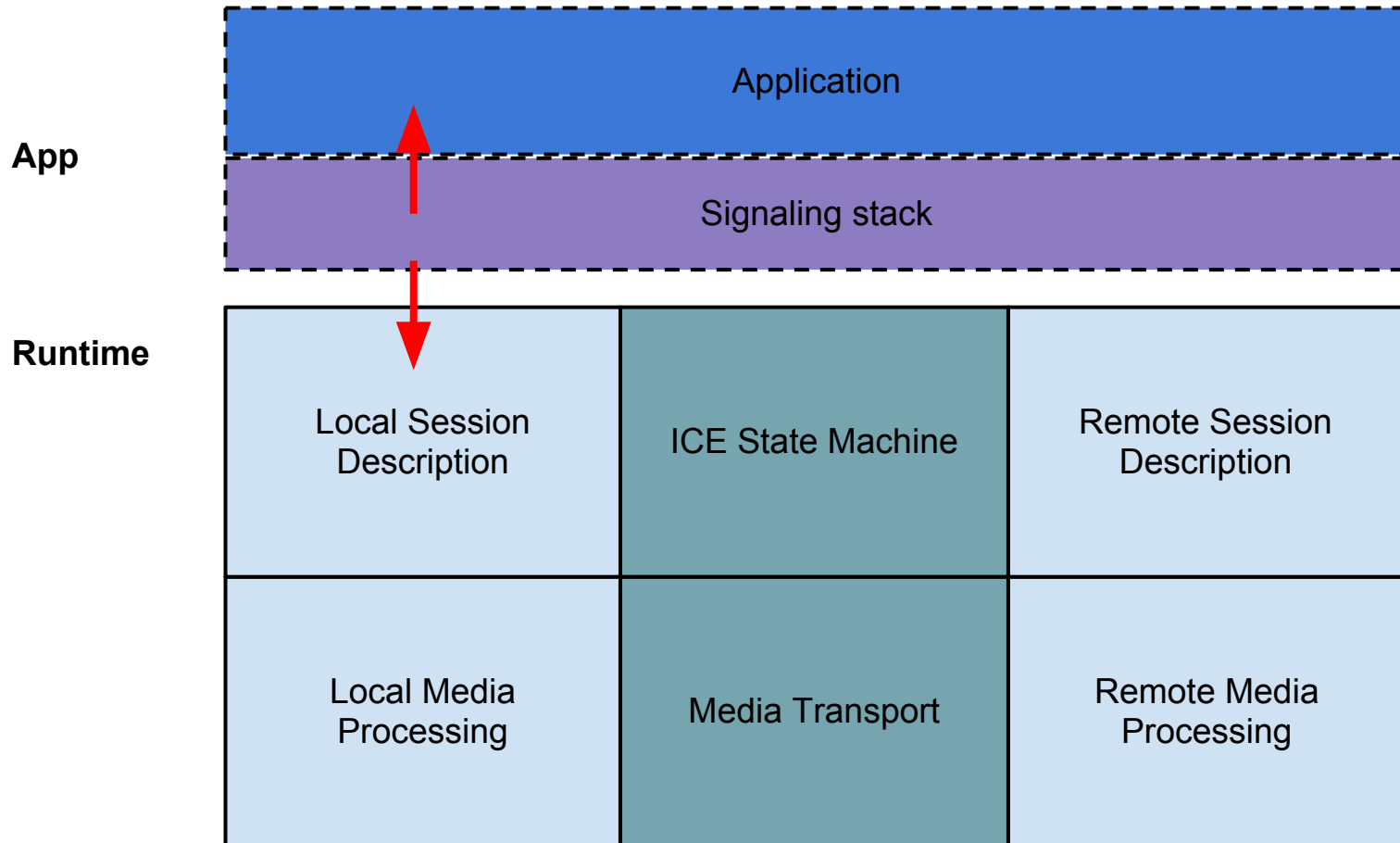
# JSEP App Diagram



# Call Setup: Offer

```
pc = new PeerConnection();  
pc.addStream(localStream, null);  
offer = pc.createOffer(null);  
pc.setLocalDescription(SDP_OFFER, offer);  
signalSocket.send(MakeInitiate(offer));
```

# Call Setup: Offer

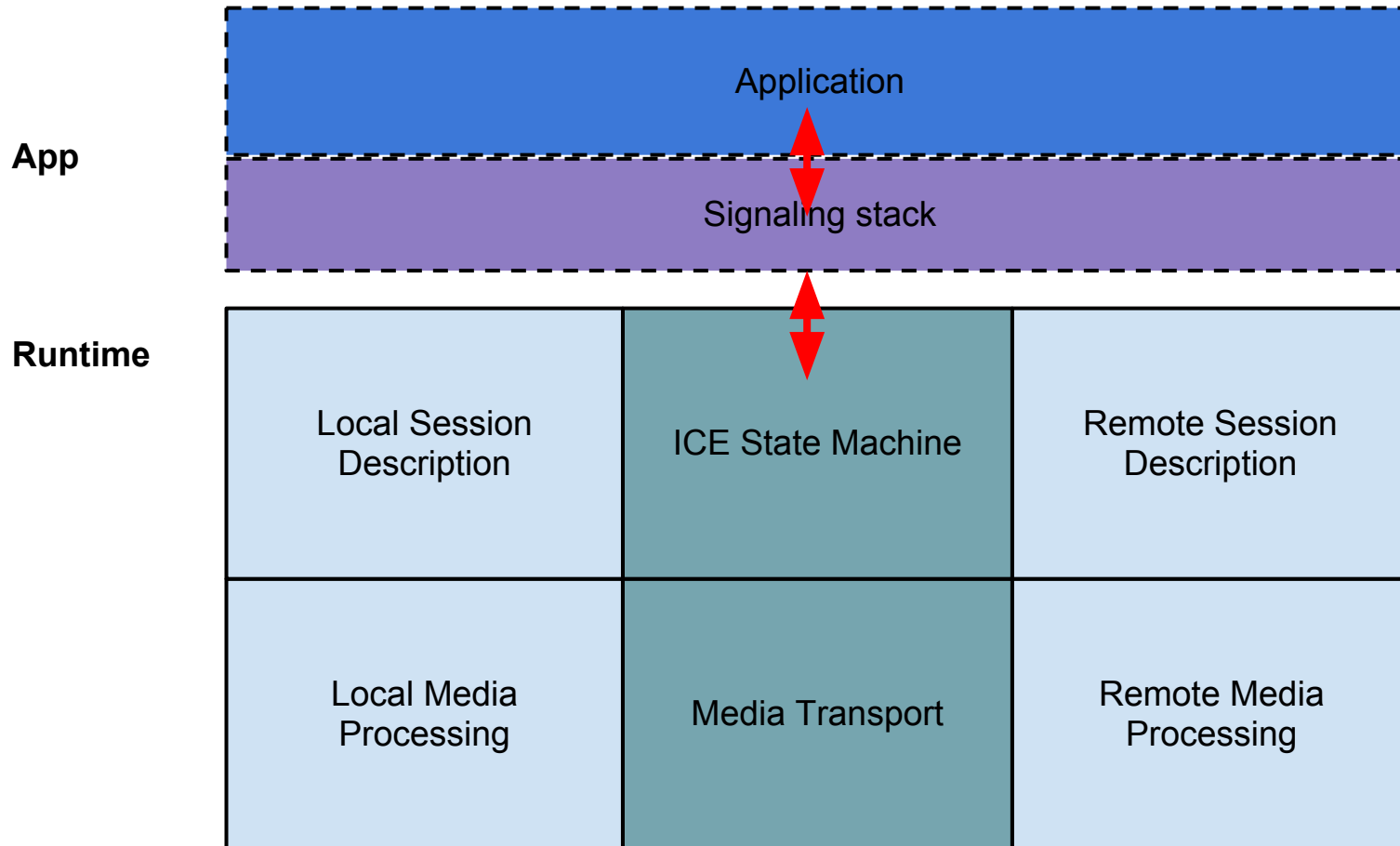




# Call Setup: Starting ICE

```
pc.startIce();  
iceCallback(media, transportInfo);  
signalSocket.send(MakeTransportInfo(  
    media, transportInfo));  
  
// later  
signalSocket.onmessage(transportInfo);  
pc.processIceMessage(transportInfo.media,  
    transportInfo.data);
```

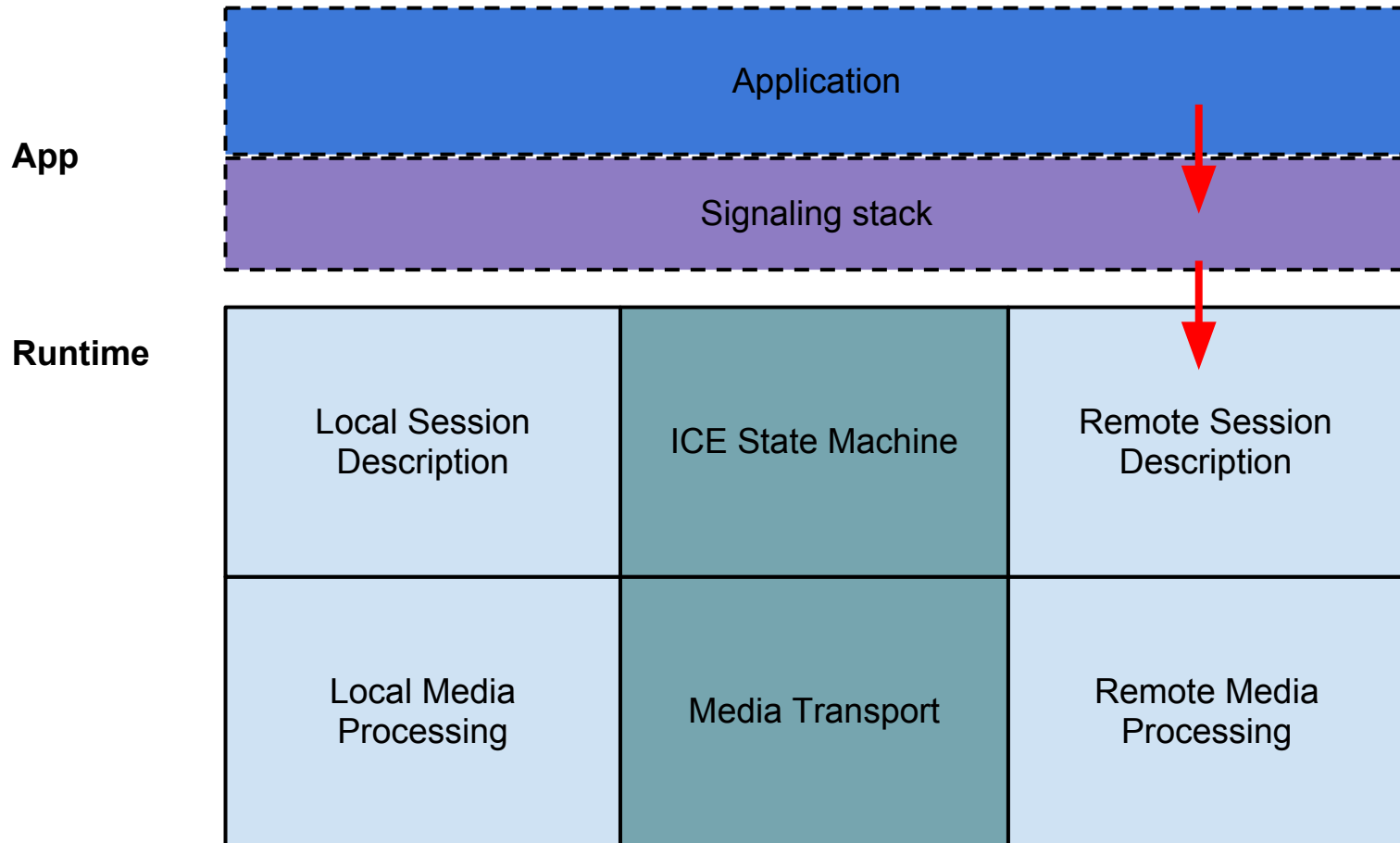
# Call Setup: Starting ICE



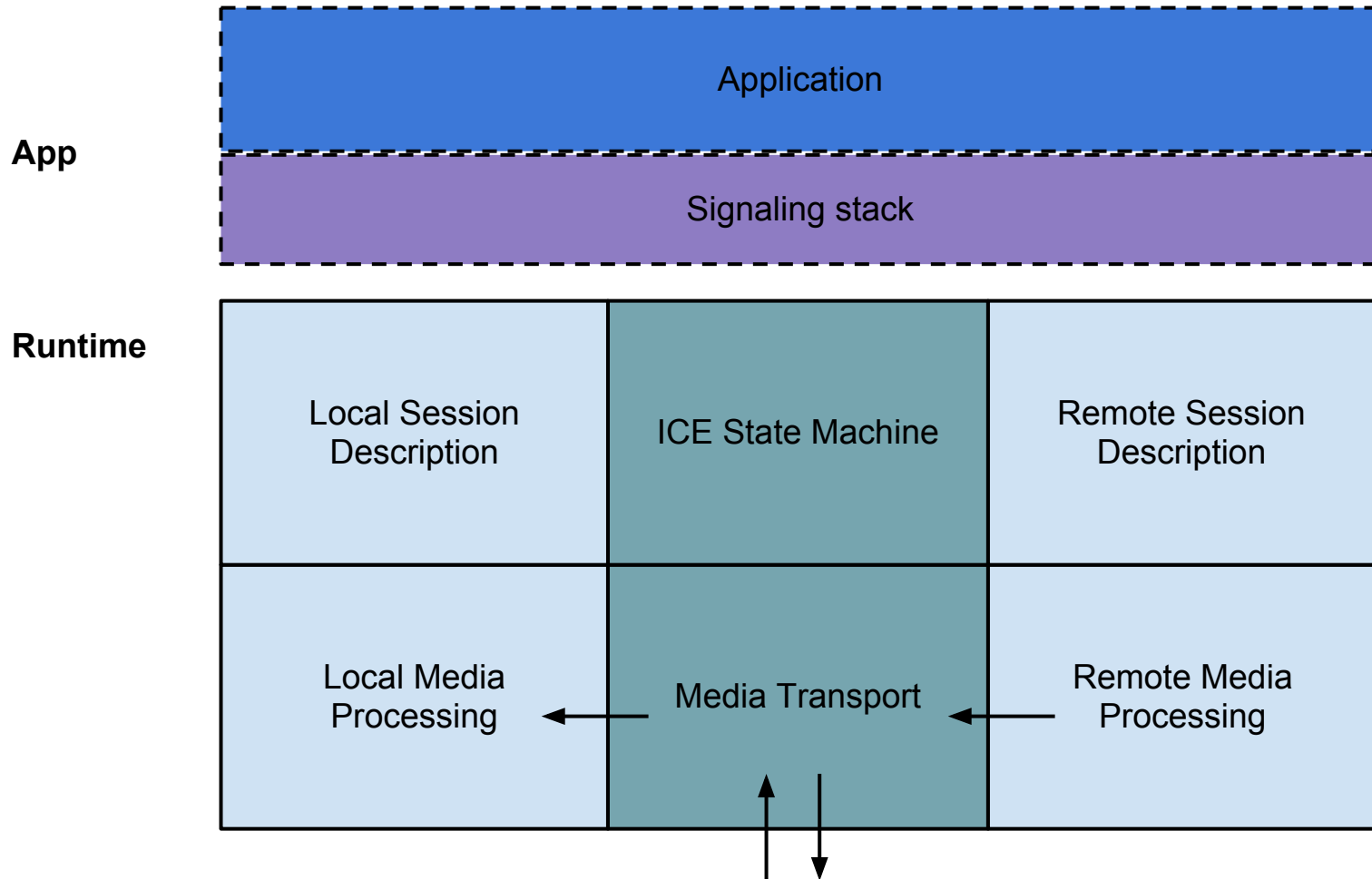
# Call Setup: Answer

```
onmessage (accept) ;  
answer = ParseAccept (accept) ;  
pc.setRemoteDescription (ANSWER, answer) ;  
onaddstream (remoteStream) ;  
onopen () ;
```

# Call Setup: Answer



# Active Call



# Call Update: Add Stream

```
pc.addStream(localStream2) ;  
offer = pc.createOffer(null) ;  
pc.setLocalDescription(SDP_OFFER, offer) ;  
signalSocket.send(MakeUpdate(offer)) ;  
...  
onmessage(accept) ;  
answer = ParseAccept(accept) ;  
pc.setRemoteDescription(SDP_ANSWER, answer) ;
```

# Call Update: Glareless Add

```
pc.addStream(localStream2) ;  
offer = pc.createOffer(null) ;  
delta = Diff(pc.localDescription, offer2) ;  
pc.setLocalDescription(SDP_OFFER, offer2) ;  
signalSocket.send(MakeStreamAdd(delta)) ;  
...  
onmessage(remoteDelta) ;  
onmessage(ackOffer2) ;  
pc.setRemoteDescription(SDP_ANSWER,  
    MakeDesc(remoteDelta)) ;  
signalSocket.send(MakeAck(remoteOffer)) ;
```

# Call Update: Hold

```
offer = pc.createOffer(null) ;  
offer = AppendSendOnly(offer) ;  
pc.setLocalDescription(SDP_OFFER, offer) ;  
signalSocket.send(MakeHold(offer)) ;
```



# Things You Can Do

- Send candidates as they are gathered
- Add/remove sources simultaneously
- Change session parameters at any time (with or without an O/A exchange)
- Control local session description that is generated and sent
- Rehydrate a session from stored state

# Impl Considerations

# New APIs: Creating SDP

## **createOffer(hints)**

Creates a session description based on the current local media state; |hints| allows for some customization. Does not reserve resources, or change state.

## **createAnswer(offer, hints)**

Like createOffer, but uses |offer| as input to create a compatible session description.

# New APIs: Hints

Hints are shortcuts to allow customization of generated offers/answers

Example: Only have audio sources, but want to receive video from remote side; pass in MediaHints with `has_video` set to true to add a `m=video` section with no sources

# New APIs: Applying SDP

## **setLocalDescription(type, desc)**

Applies the local description, e.g. recv codecs, encryption keys. Changes state.

## **setRemoteDescription(type, desc)**

Applies the remote description, e.g. send codecs, decryption keys.

Throws exception on invalid state or params

# New APIs: ICE

## **IceCallback(media, transportInfo)**

Callback function that receives transport information that needs to be signaled

## **processIceMessage(media, transportInfo)**

Invoked to handle received transport information

# Message Formats

## Session Descriptions

Standard SDP

## Transport Info

a=ice-candidate lines

a=ice-ufrag, password lines

a=fingerprint (for DTLS)

a=group (for BUNDLE)

# Complexity

- JSEP does require more code (~60 lines for a basic example, w/o glare handling)
- But can be easily encapsulated within a JS library
- Moreover, this library can perform protocol translation too (e.g. convert to SIP, XMPP, or ROAP)
- Powerful API/JS libraries is consistent with overall web application trends (e.g. WebGL, IndexedDB)



# JSEP vs ROAP

## Key differences:

- Signaling mechanism lives in app/JS
- Early transport negotiation supported
- App has control over local description
- App can change session parameters at any time, without O/A if desired
- App can restore session from cached state
- More JS code, but under app control

# Real-World Benefits

- Proven model; Hangouts now using a form of this API
- Early candidate gathering improves start time by over a second in >20% of calls
- Glare conditions become a non-issue for many apps
- Features can be added without new browser APIs (e.g. one-way video, hold, res change)
- Calls can persist across application upgrades

# Questions?