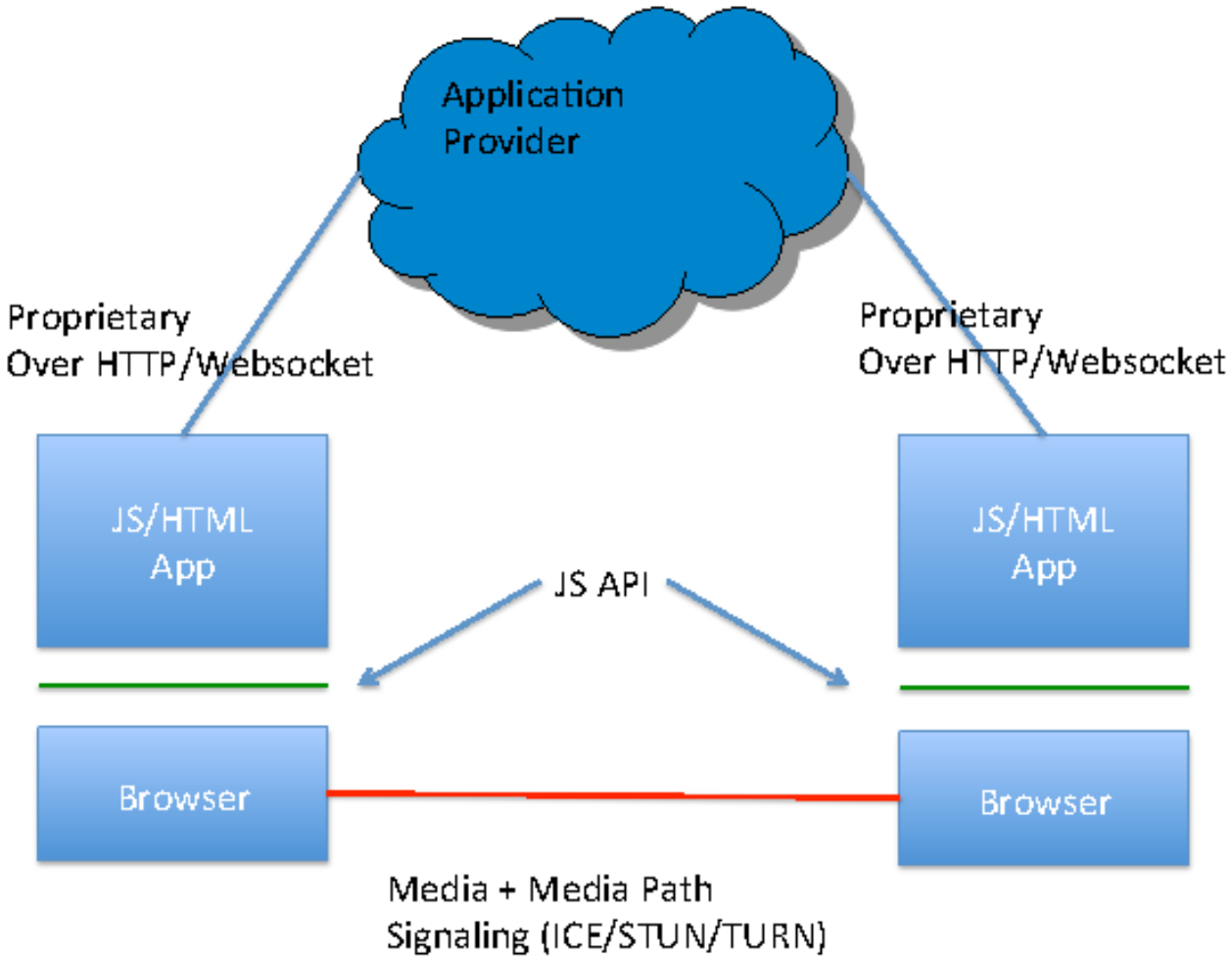
RTCWEB architecture

Harald Alvestrand

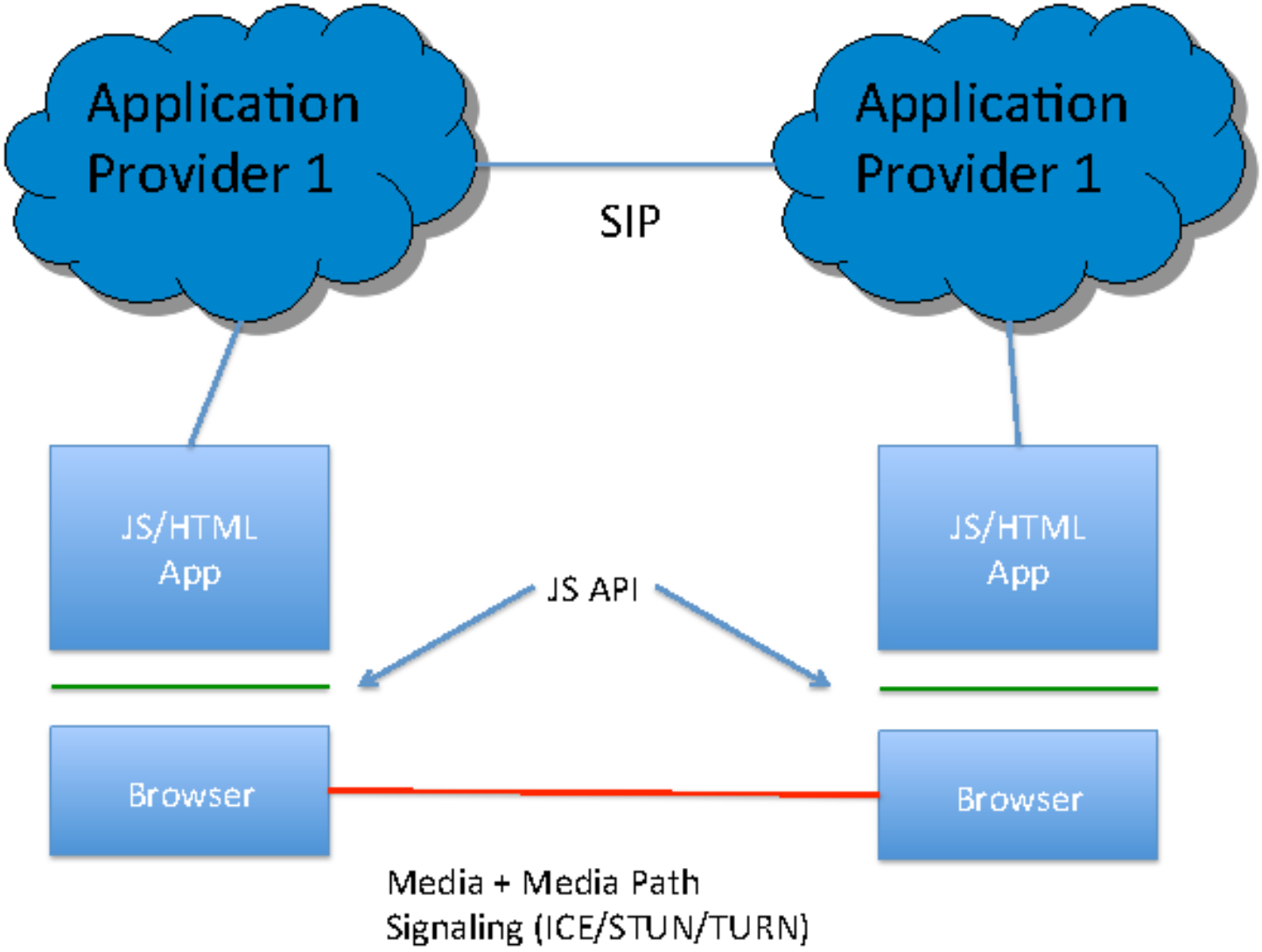
RTCWEB goals

- Real Time Communication in the Browser
  - Browser to Browser is Job Number One
  - Usable by JS applications
  - Not (just) for apps we have already deployed
- Motherhood and Apple Pie
  - Secure
  - Manageable
  - Network Friendly

Base Case (Model 1)



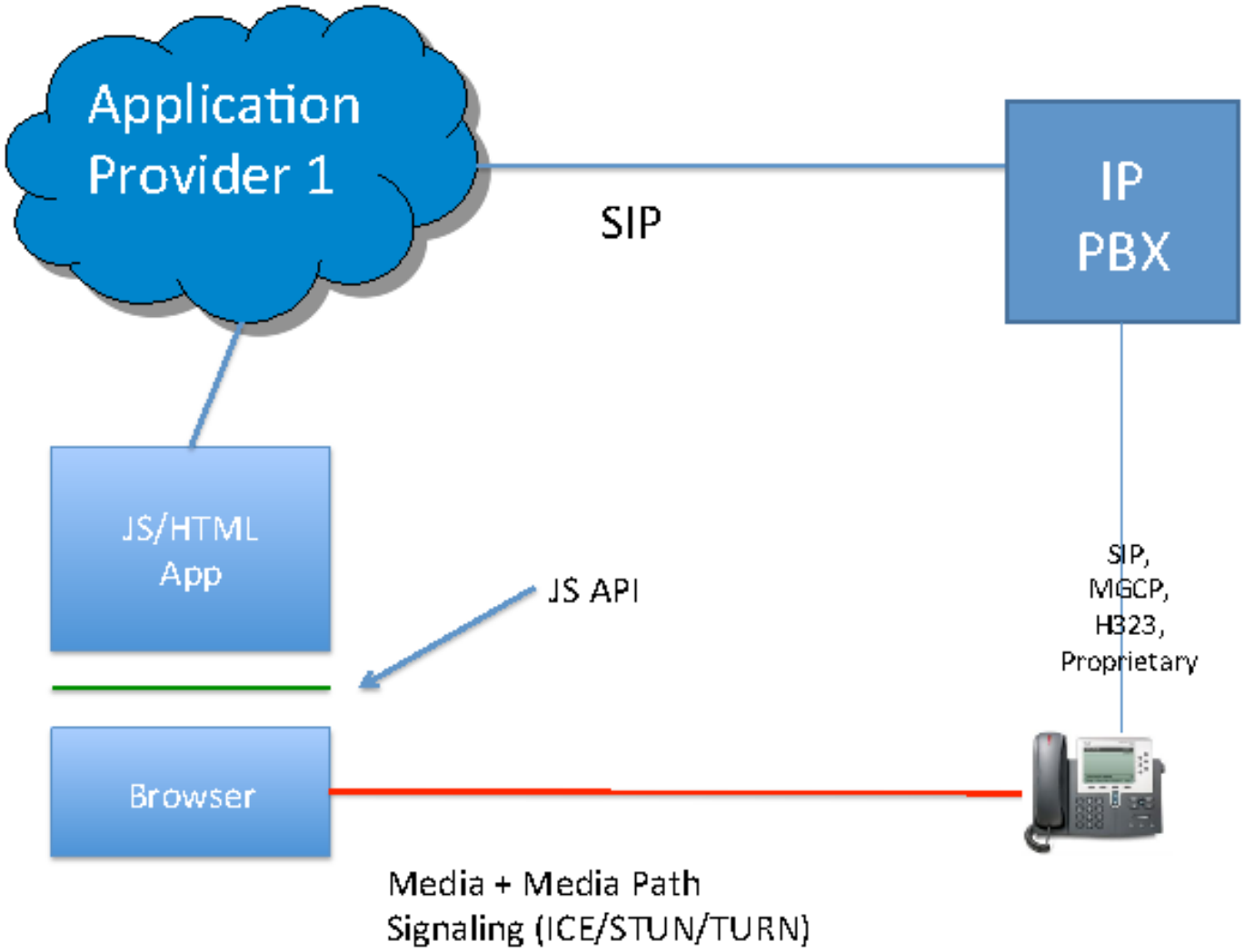
Provider Interop Case



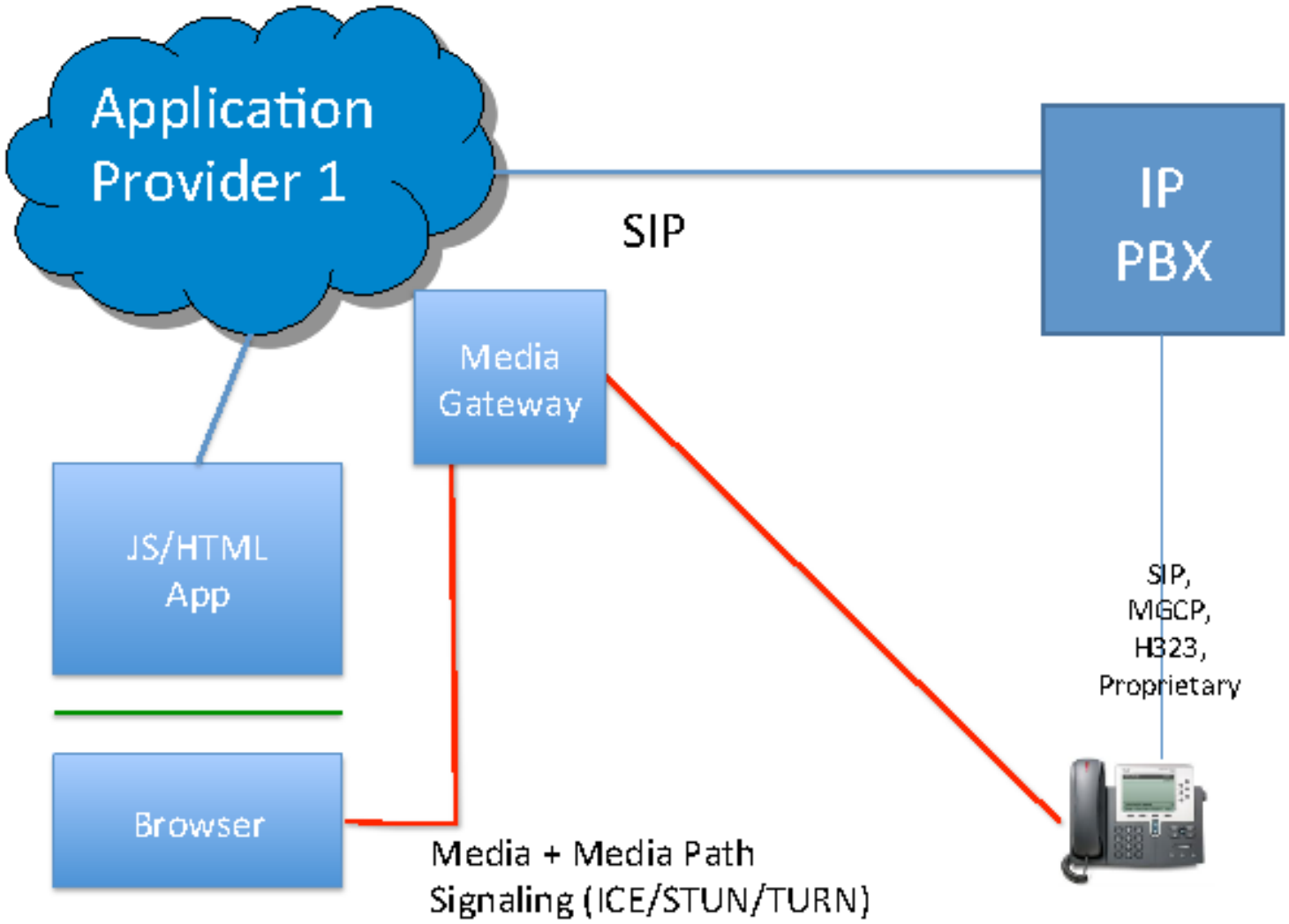
RTC is not a green field

- Lots of protocols defined
  - Code
  - Knowledge
  - Hardware
- Lots of stuff deployed
  - Proprietary
  - Using standard protocols internally
  - Interoperable

Existing Gear Case 1



Existing Gear Case 2



RTCWEB constraints

- Browser environment
  - A few, massively deployed implementations
  - Scrutiny on code complexity, security, size
- Uncontrolled LAN environment
  - All kinds of middle boxes (NAT, FW, proxy)
  - No ability to configure local network
  - Must work with zero configuration

Function layers

- Data transport
- Data framing and securing
- Data formats
- Connection management
- Presentation and control
- Local system support

draft-alvestrand-rtcweb-overview

Data transport

- IP (obvious)
- ICE for establishing desire to communicate
- Direct if possible, with TURN relay fallback
- UDP if possible, with TCP fallback

Controversy:

- Whether all of ICE is appropriate
  - Draft-kaufman-rtcweb-traversal
- Congestion control strategies (TFRC?)

Data framing

- RTP for media, using RTP/SAVPF profile
- RTCP multiplexing, symmetric RTP/RTCP
- More details

draft-cbran-rtcweb-protocols
draft-perkins-rtcweb-rtp-usage

Controversial:

- RTP multiplexing – session-per-media-stream?
- Framing of non-media data - UDP/DCCP/DTLS?

Data formats

- MTI Codec selection
  - OPUS (may be uncontroversial)
  - At least one low-quality “phone” audio codec

Controversial:

- Video codec status (do not discuss now)
- Telephone events (important use case?)
- Formats for non-media data

Connection management

- Clear requirement to negotiate connections
  - Legacy has SDP as “lingua franca of descriptions”
  - SDP has lots of legacy with it
- No dominant accepted management protocol
  - Large SIP installed base – limited interoperability
  - XMPP has separate following
  - Simple use cases may be better off on proprietary
  - Gateway solutions better on browser footprint?

Presentation and Control

- User control over his own devices
- Management of streams locally
- May interact with stream control, but mostly a local matter – may be better addressed in W3C?

Local system support functions

Such as:

- Echo cancellation
- Automatic gain control
- Camera zoom/pan/tilt

Some aspects are inter-user and may need standardization.
Others are matters of “good citizenship” (“mute when not speaking in large meetings”).

For now, leave it for later.

Summary

- Consider the environment!
- We have lots of things that are obvious
  - Some of these have implementations
- We have a few known controversies
  - Need to figure out if one alternative has “rough consensus” and move forward
  - If no rough consensus – what do we need to test?
- We will find more controversies over time

Rough consensus and running code.