WebRTC Call Flows

W3C WebRTC Interim Meeting

Boston, MA

February 6, 2013

Goals

- Walk through examples in spec
- Note disagreements on “what happens here”
- Note questions about “what happens here if….”
- Don’t make decisions - list questions

10.1 Simple Peer-to-peer Example

When two peers decide they are going to set up a connection to each other, they both go through these steps. The STUN/TURN server configuration describes a server they can use to get things like their public IP address or to set up NAT traversal. They also have to send data for the signaling channel to each other using the same out-of-band mechanism they used to establish that they were going to communicate in the first place.

10.1, cont

```
var signalingChannel = new SignalingChannel();
var configuration = { "iceServers": [{ "url": "stun:stun.example.org" }] };
var pc;

// call start() to initiate
function start() {
    pc = new RTCPeerConnection(configuration);

    // send any ice candidates to the other peer
    pc.onicecandidate = function (evt) {
        if (evt.candidate)
            signalingChannel.send(JSON.stringify({ "candidate": evt.candidate }));
    };

    // let the "negotiationneeded" event trigger offer generation
    pc.onnegotiationneeded = function () {
        pc.createOffer(localDescCreated, logError);
    }

    // once remote stream arrives, show it in the remote video element
    pc.onaddstream = function (evt) {
        remoteView.src = URL.createObjectURL(evt.stream);
    };

    // get a local stream, show it in a self-view and add it to be sent
    navigator.getUserMedia({ "audio": true, "video": true }, function (stream) {
        selfView.src = URL.createObjectURL(stream);
        pc.addStream(stream);
    });
}
```

10.1, cont

```
function localDescCreated(desc) {
    pc.setLocalDescription(desc, function () {
        signalingChannel.send(JSON.stringify({ "sdp": pc.localDescription }));
    }, logError);
}

signalingChannel.onmessage = function (evt) {
    if (!pc)
        start();

    var message = JSON.parse(evt.data);
    if (message.sdp)
        pc.setRemoteDescription(new RTCSessionDescription(message.sdp), function () {
            // if we received an offer, we need to answer
            if (pc.remoteDescription.type == "offer")
                pc.createAnswer(localDescCreated, logError);
        }, logError);
    else
        pc.addIceCandidate(new RTCIceCandidate(message.candidate));
};

function logError(error) {
    log(error.name + ": " + error.message);
}
```

10.3 Peer-to-peer Data Example

This example shows how to create a RTCDataChannel object and perform the offer/answer exchange required to connect the channel to the other peer. The RTCDataChannel is used in the context of a simple chat application and listeners are attached to monitor when the channel is ready, messages are received and when the channel is closed.

10.3, cont

```
var signalingChannel = new SignalingChannel();
var configuration = { "iceServers": [{ "url": "stun:stun.example.org" }] };
var pc;
var channel;

// call start(true) to initiate
function start(isInitiator) {
    pc = new RTCPeerConnection(configuration);

    // send any ice candidates to the other peer
    pc.onicecandidate = function (evt) {
        if (evt.candidate)
            signalingChannel.send(JSON.stringify({ "candidate": evt.candidate }));
    };

    // let the "negotiationneeded" event trigger offer generation
    pc.onnegotiationneeded = function () {
        pc.createOffer(localDescCreated, logError);
    }

    if (isInitiator) {
        // create data channel and setup chat
        channel = pc.createDataChannel("chat");
        setupChat();
    } else {
        // setup chat on incoming data channel
        pc.ondatachannel = function (evt) {
            channel = evt.channel;
            setupChat();
        };
    }
}
```

10.3, cont

```
function localDescCreated(desc) {
    pc.setLocalDescription(desc, function () {
        signalingChannel.send(JSON.stringify({ "sdp": pc.localDescription }));
    }, logError);
}

signalingChannel.onmessage = function (evt) {
    if (!pc)
        start(false);

    var message = JSON.parse(evt.data);
    if (message.sdp)
        pc.setRemoteDescription(new RTCSessionDescription(message.sdp), function () {
            // if we received an offer, we need to answer
            if (pc.remoteDescription.type == "offer")
                pc.createAnswer(localDescCreated, logError);
        }, logError);
    else
        pc.addIceCandidate(new RTCIceCandidate(message.candidate));
};

function setupChat() {
    channel.onopen = function () {
        // e.g. enable send button
        enableChat(channel);
    };

    channel.onmessage = function (evt) {
        showChatMessage(evt.data);
    };
}
```

10.3, cont

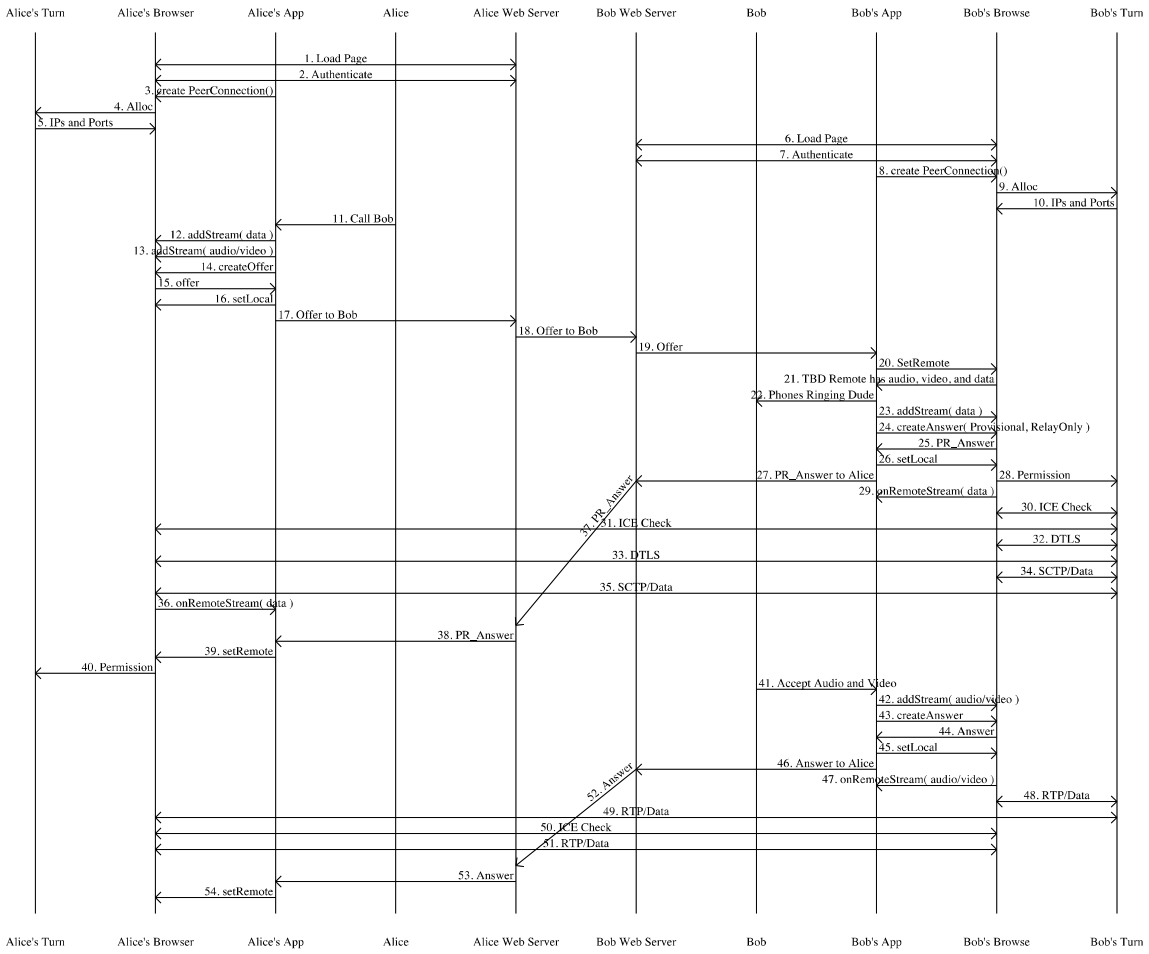
```
function sendChatMessage(msg) {
    channel.send(msg);
}

function logError(error) {
    log(error.name + ": " + error.message);
}
```

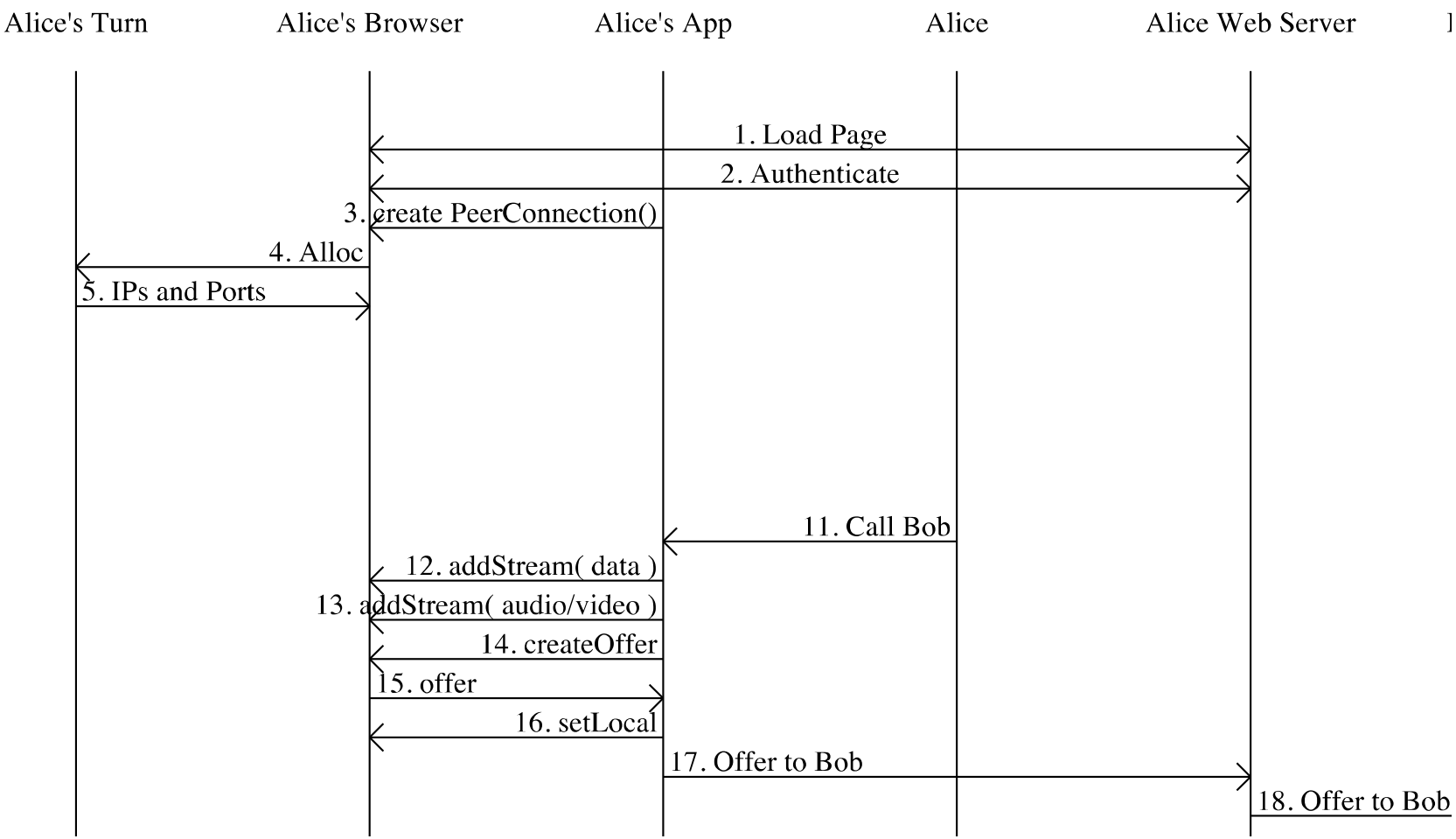
10.4 Call Flow Browser to Browser

- Editors' Note: This example flow needs to be discussed on the list and is likely wrong in many ways.
- This shows an example of one possible call flow between two browsers. This does not show every callback that gets fired but instead tries to reduce it down to only show the key events and messages.

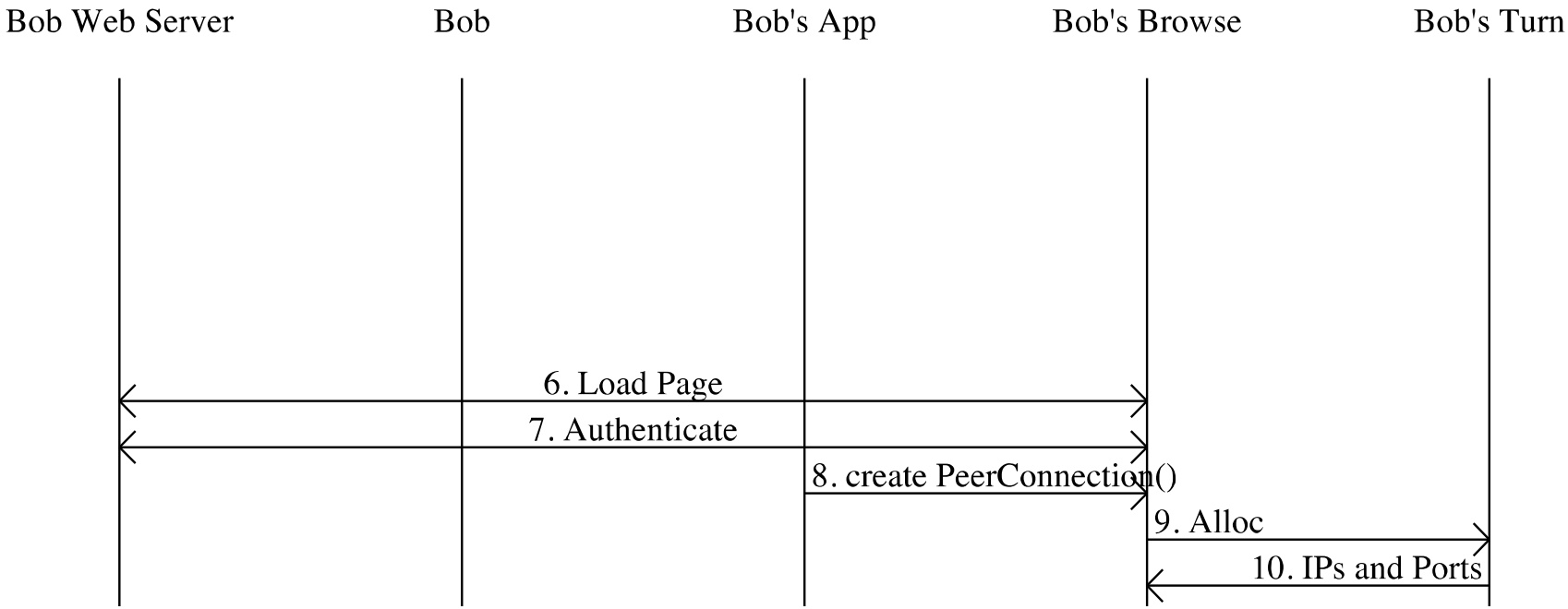
10.4, cont


Alice's Turn
Alice's Browser
Alice's App
Alice
Alice Web Server
Bob Web Server
Bob
Bob's App
Bob's Browse
Bob's Turn
1. Load Page
2. Authenticate
3. create PeerConnection()
4. Alloc
5. IPs and Ports
6. Load Page
7. Authenticate
8. create PeerConnection()
9. Alloc
10. IPs and Ports
11. Call Bob
12. addStream(data)
13. addStream(audio/video)
14. createOffer
15. offer
16. setLocal
17. Offer to Bob
18. Offer to Bob
19. Offer
20. SetRemote
21. TBD Remote has audio, video, and data
22. Phones Ringing Dude
23. addStream(data)
24. createAnswer(Provisional, RelayOnly)
25. PR_Answer
26. setLocal
27. PR_Answer to Alice
28. Permission
29. onRemoteStream(data)
30. ICE Check
37. PR_Answer
31. ICE Check
32. DTLS
33. DTLS
34. SCTP/Data
35. SCTP/Data
36. onRemoteStream(data)
38. PR_Answer
39. setRemote
40. Permission
41. Accept Audio and Video
42. addStream(audio/video)
43. createAnswer
44. Answer
45. setLocal
46. Answer to Alice
47. onRemoteStream(audio/video)
48. RTP/Data
52. Answer
49. RTP/Data
50. ICE Check
51. RTP/Data
53. Answer
54. setRemote

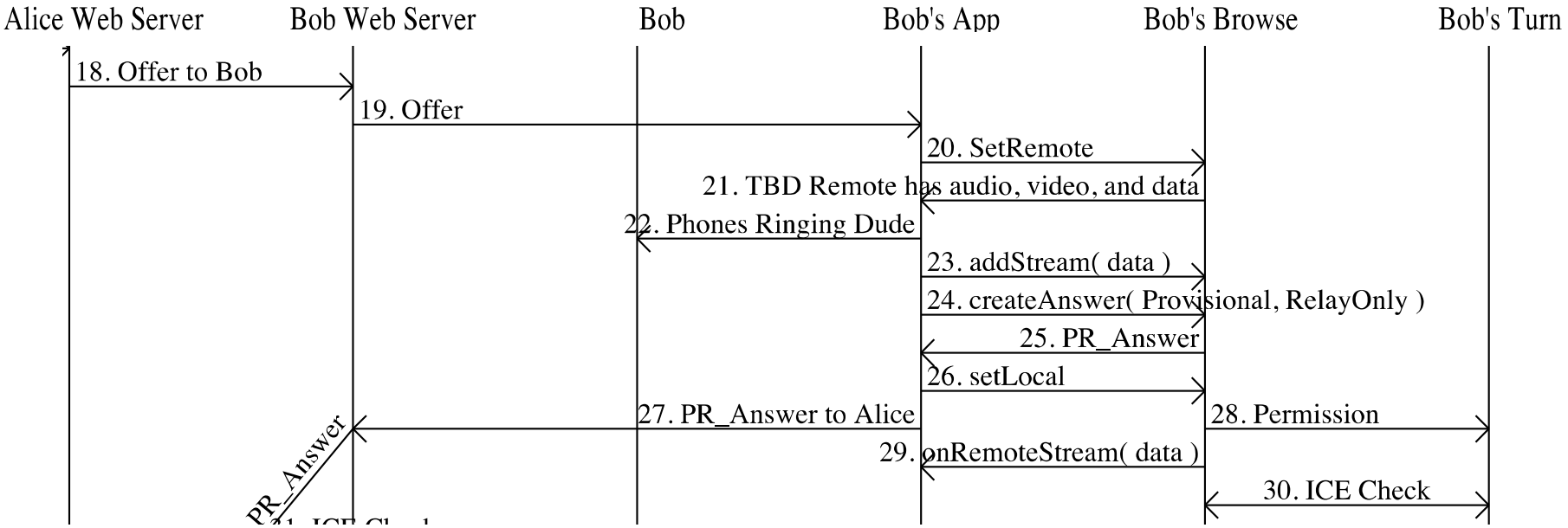
10.4, cont


Alice's Turn
Alice's Browser
Alice's App
Alice
Alice Web Server
1. Load Page
2. Authenticate
3. create PeerConnection()
4. Alloc
5. IPs and Ports
11. Call Bob
12. addStream(data)
13. addStream(audio/video)
14. createOffer
15. offer
16. setLocal
17. Offer to Bob
18. Offer to Bob

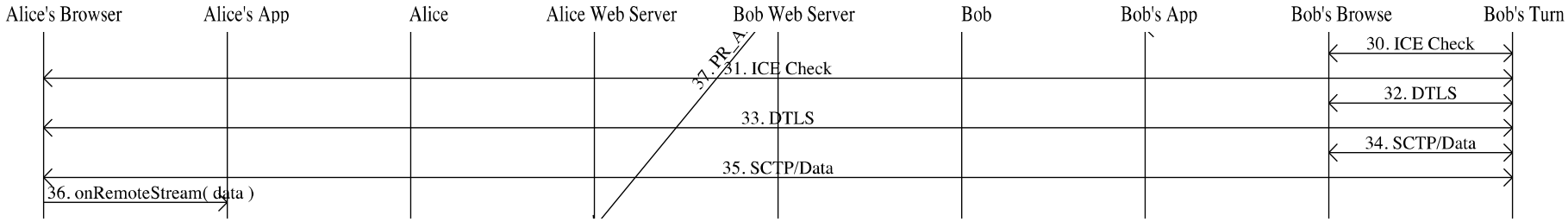
10.4, cont


Bob Web Server
Bob
Bob's App
Bob's Browse
Bob's Turn
6. Load Page
7. Authenticate
8. create PeerConnection()
9. Alloc
10. IPs and Ports

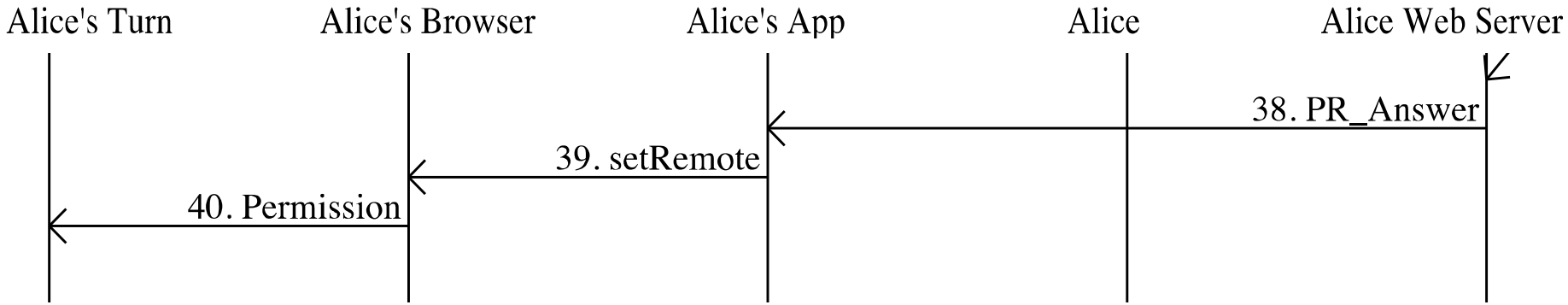
10.4, cont


Alice Web Server
Bob Web Server
Bob
Bob's App
Bob's Browse
Bob's Turn
18. Offer to Bob
19. Offer
20. SetRemote
21. TBD Remote has audio, video, and data
22. Phones Ringing Dude
23. addStream(data)
24. createAnswer(Provisional, RelayOnly)
25. PR_Answer
26. setLocal
27. PR_Answer to Alice
28. Permission
29. onRemoteStream(data)
30. ICE Check
PR_Answer

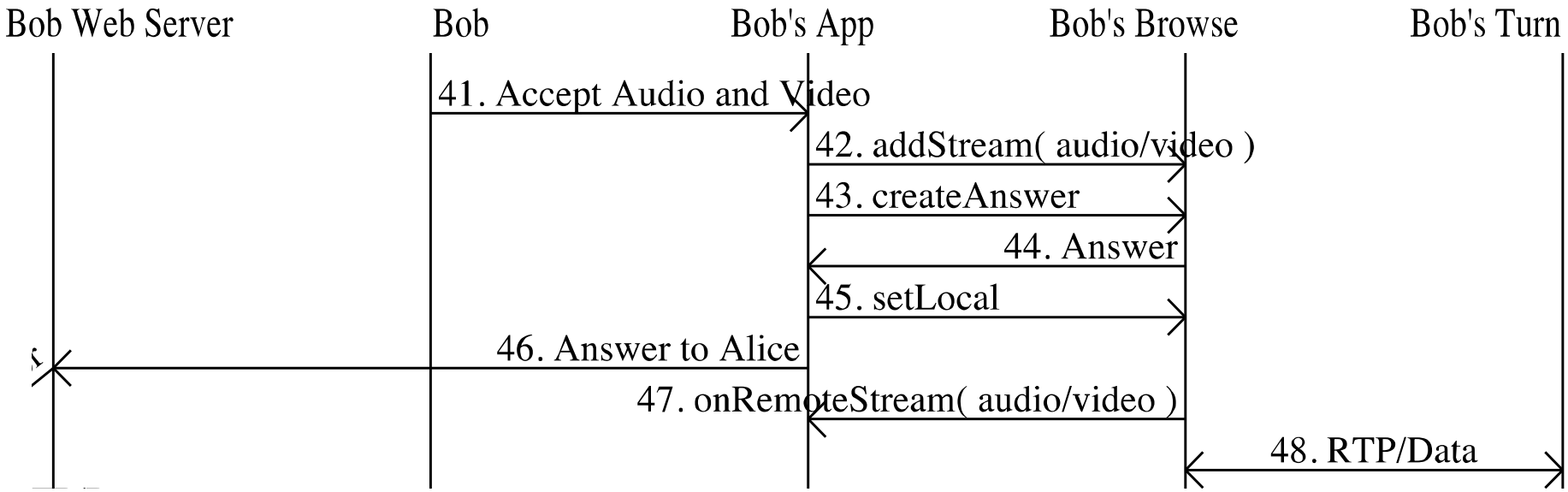
10.4, cont


Alice's Browser
Alice's App
Alice
Alice Web Server
Bob Web Server
Bob
Bob's App
Bob's Browse
Bob's Turn
30. ICE Check
37. PR_A
31. ICE Check
32. DTLS
33. DTLS
34. SCTP/Data
35. SCTP/Data
36. onRemoteStream(data)

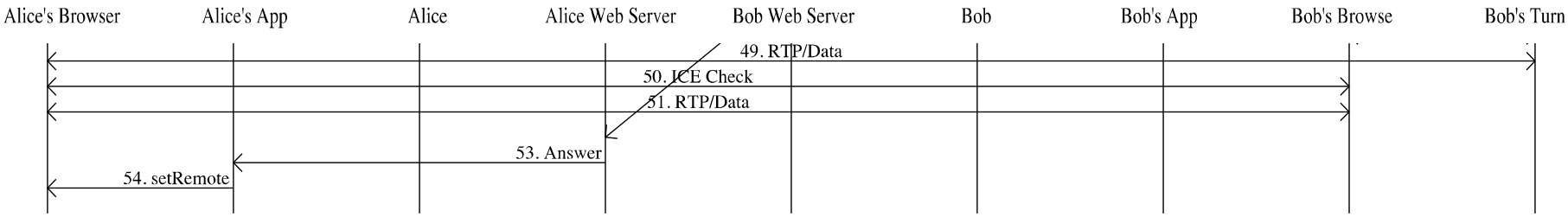
10.4, cont


Alice's Turn
Alice's Browser
Alice's App
Alice
Alice Web Server
38. PR_Answer
39. setRemote
40. Permission

10.4, cont


Bob Web Server
Bob
Bob's App
Bob's Browse
Bob's Turn
41. Accept Audio and Video
42. addStream(audio/video)
43. createAnswer
44. Answer
45. setLocal
46. Answer to Alice
47. onRemoteStream(audio/video)
48. RTP/Data

10.4, cont


Alice's Browser
Alice's App
Alice
Alice Web Server
Bob Web Server
Bob
Bob's App
Bob's Browse
Bob's Turn
49. RTP/Data
50. ICE Check
51. RTP/Data
53. Answer
54. setRemote

10.5 DTMF Example

Sending the DTMF signal “1234” with 500 ms per tone:

```
sender = pc.createDTMFSender(track);
if (sender.canSendDTMF) {
  sender.insertDTMF(“1234”, 500);
} else {
  alert(‘DTMF function not available’);
}
```

10.5, cont

Sending the DTMF signal “1234”, and lighting up a key using “lightKey(x)” while the tone is playing (assuming that lightKey(‘’) will darken all the keys):

```
sender = pc.createDTMFSender(track);
sender.ontonechange = function(e) {
  lightKey(e.tone);
}
sender.insertDTMF(‘1234’);
```

10.5, cont

Sending a 1-second “1” tone followed by a 2-second “2” tone:

```
sender = pc.createDTMFSender(track);
sender.ontonechange = function(e) {
  if (e.tone == ‘’) {
    sender.insertDTMF(‘2’, 2000);
  }
}
sender.insertDTMF(‘1’, 1000);
```

10.5, cont

Sending the tone string ‘12345’, and appending the tone string ‘6789’ before the tone finishes playing:

```
sender = pc.createDTMFSender(track);
sender.insertDTMF(‘12345’);
// Other things happen.....
sender.insertDTMF(sender.toneBuffer + ‘6789’);
```