W3C ORTC Community Group Meeting #9

June 24, 2015 10:00AM-11:30AM PDT

Chair: Erik Lagerway (erik@hookflash.com)

W3C CG IPR Policy

- See the Community License Agreement for details.
- Goals are
  - Enable rapid spec development
  - Safe to implement via royalty-free commitments from participants+employers
  - Comfort for committers by limiting scope to OWN contributions
  - Transparency about who is making commitments
- How it works in practice
  - Anyone can post to public-ortc@w3.org
  - CG members who have signed CLA can post to public-ortc-contrib
  - Editor should ensure that spec includes only “contributions”, CC-ing public-ortc-contrib makes that easier on the editor.

Welcome!

- Welcome to the 9th meeting of the W3C ORTC Community Group! Now 106 members!
- During this meeting, we hope to:
  - Bring you up to date on the status of the ORTC specification
  - Make progress on some outstanding issues
  - Organize/plan for implementation feedback

About this Virtual Meeting

Information on the meeting:

- Hangouts Meeting
  - Web Broadcast Link (**view only**)
  - Participatory Hangout Link (when meeting starts)
- Link to Slides has been published on CG home page & ORTC.org
- Scribe? IRC http://irc.w3.org/ Channel: ORTC
- The meeting is being recorded.

W3C ORTC Community Group Basics

- W3C ORTC CG website:
  - http://www.w3.org/community/ortc/
- Public mailing list: public-ortc@w3.org
  - Join Here - link on the right hand side
  - Non-members can post to this list.
  - Non-member contributions are problematic.
- Contributor's mailing list: public-ortc-contrib@w3.org
  - Join Here - link on the right hand side
  - Members only, preferred list for contributions to the specification.

Associated Sites

- ORTC developer website: http://ortc.org/
  - Editor’s drafts, pointers to github repos, etc.
- ORTC API Issues List: https://github.com/openpeer/ortc/issues?state=open

Editor’s Draft Changes

- June 2015 Editor’s draft:
  - http://ortc.org/wp-content/uploads/2015/06/ortc.html

Changes from the 07 May Editor’s Draft:

Phillip Hancke’s Review (Issue 198***).***

- Editorial fixes (next slide)
- Errors in sample code
  - Trivial: mismatched }’s and ;’s
  - Not so trivial: forking problems in the API (more later)
    - Sample code now points out the problem via comments.

Editorial: Philipp Hancke's Review (Issue 198)

a. Introduction rewritten to better explain the relationship between the objects.
b. Sections re-ordered to go from bottom of the stack towards the top.
    i. Overview
    ii. The RTCIceGatherer Object
    iii. The RTCIceTransport Object
    iv. The RTCDtlsTransport Object
    v. The RTCRtpSender Object
    vi. The RTCRtpReceiver Object
    vii. The RTCIceTransportController Object

Philipp Hancke's Review (cont'd)

- New Issues created for:
  - No "Failed" state in IceTransportState (Issue 199)
  - Error handling when calling getStats on "closed" objects (Issue 214).
  - RTCIceCandidate: Support for "Generation" (Issue 212)
  - Privacy Issues (Issue 213).
  - Handling of certificates and fingerprints in the non-RTP/RTCP mux case (Issue 210)
  - DtlsTransport.getLocalParameters and certificate creation (Issue 211)

Editor’s Draft Changes

ICE

- Issue 199: Added the "failed" state to **RTCIceTransportState**
- Issue 207: Added a complete attribute to the **RTCIceCandidateComplete** dictionary
- Issue 208: Updated the description of RTCIceGatherer.close() and the "closed" state
- Issue 216: Clarified ICE state transitions due to consent failure

RTCWEB/WebRTC 1.0 compatibility

- Issue 214: Updated Statistics API error handling to reflect proposed changes to the WebRTC 1.0 API
- Issue 215: Updated Section 10 (RTCDtmfSender) to reflect changes in the WebRTC 1.0 API
- Issue 217: Added a reference to draft-ietf-rtcweb-fec

RTCIceCandidateComplete (Issue 207)

May Editor's draft:

```
typedef (RTCIceCandidate or RTCIceCandidateComplete) RTCIceGatherCandidate;
dictionary RTCIceCandidateComplete {
};
```

Proposed change:

```
typedef (RTCIceCandidate or RTCIceCandidateComplete) RTCIceGatherCandidate;
dictionary RTCIceCandidateComplete {
        boolean complete = true;
};
```

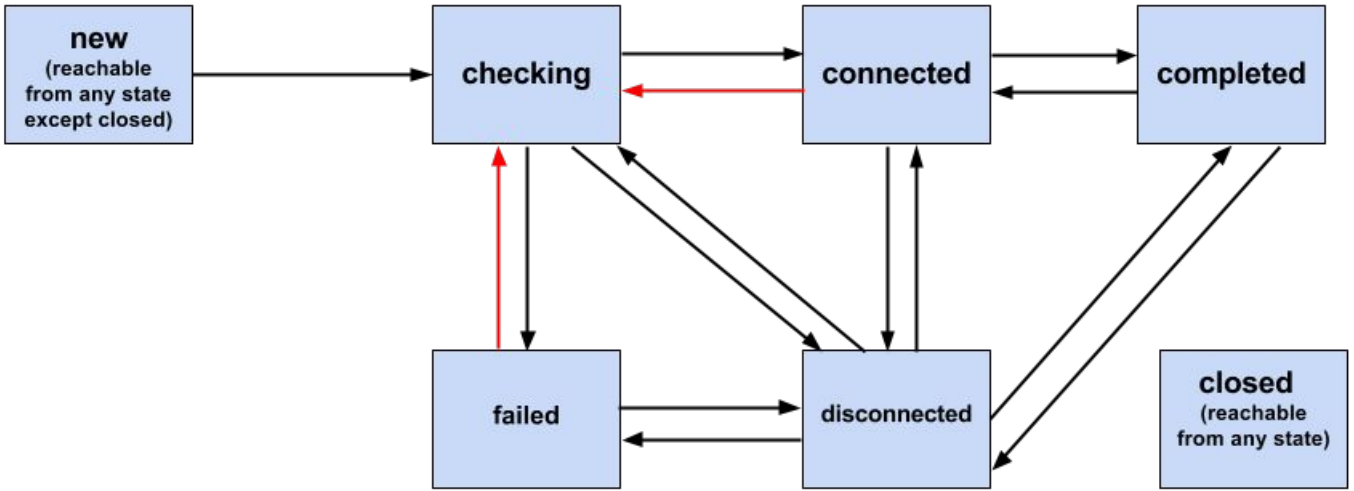
Justin: This seems kind of bizarre, an event with a variable that is only set to a single value. Can't this be handled entirely through IceGathererStateChange?

Robin: IceGatherStateChange tells the local peer that "candidates are complete". The question is how that information is signaled over the wire and then provided in RTCIceTransport.addRemoteCandidate(). The proposal is to use the RTCIceCandidateComplete dictionary for that purpose, so that state changes are triggered by calling RTCIceTransport.addRemoteCandidate(RTCIceCandidateComplete). Having one attribute seems cleaner than signaling null and passing that to addRemoteCandidate.

Revised *RTCIceTransportState* Definitions

new	The **RTCIceTransport** object is waiting for remote candidates to be supplied. In this state the object can respond to incoming connectivity checks.
checking	The **RTCIceTransport** has received at least one remote candidate, and a local and remote **RTCIceCandidateComplete** dictionary was not added as the last candidate. In this state the **RTCIceTransport** is checking candidate pairs but has not yet found a successful candidate pair, or liveness checks have failed (such as those in [CONSENT]) on a previously successful candidate pair.
connected	The **RTCIceTransport** has received a response to an outgoing connectivity check, or has received incoming DTLS/media after a successful response to an incoming connectivity check, but is still checking other candidate pairs to see if there is a better connection. In this state outgoing media is permitted.
completed	A local and remote **RTCIceCandidateComplete** dictionary was added as the last candidate to the **RTCIceTransport** and all appropriate candidate pairs have been tested and at least one functioning candidate pair has been found.
disconnected	The **RTCIceTransport** has received at least one local and remote candidate, and a local and remote **RTCIceCandidateComplete**dictionary was not added as the last candidate, but all appropriate candidate pairs thus far have been tested and failed (or consent checks [CONSENT], once successful, have now failed). Other candidate pairs may become available for testing as new candidates are trickled, and therefore the "failed" state has not been reached.
failed	A local and remote RTCIceCandidateComplete dictionary was added as the last candidate to the **RTCIceTransport** and all appropriate candidate pairs have been tested and failed.
closed	The **RTCIceTransport** has shut down and is no longer responding to STUN requests.

Revised ICE State Diagram (199, 216)



Red = additional transitions due to consent loss (connected -> checking) and pending decisions around "what is continuous nomination" (failed -> checking)

Statistics API Error Handling (Issue 214)

Goal: same behavior as in WebRTC 1.0. But what is that?

11 June Editor's draft (http://w3c.github.io/webrtc-pc/#statistics-model) Section 8.2.1:

1. If the **RTCPeerConnection** object's RTCPeerConnection signalingState is closed, throw an InvalidStateError exception.

However, there is an open issue to return the last stats object instead:
https://github.com/w3c/webrtc-stats/issues/3
Also, a Firefox bug:
https://bugzilla.mozilla.org/show_bug.cgi?id=1056433

Statistics API Error Handling (cont'd)

The May ORTC Editor's draft sez (in Section 13.1):

1. For RTCDtlsTransport.getStats(), check whether RTCDtlsTransport.start() has been called; if not, throw an InvalidStateError exception. For RTCIceTransport.getStats(), check whether RTCIceTransport.start() has been called; if not, or if RTCIceTransport.stop() has been called, throw an InvalidStateError exception. For RTCRtpSender.getStats(), check whether RTCRtpSender.send(parameters) has been called; if not, throw anInvalidStateError exception. For RTCRtpReceiver.getStats(), check whether RTCRtpReceiver.receive(parameters) has been called; if not, throw anInvalidStateError exception.

Why??

- If objects have not been "started", can't we just return appropriate objects (e.g. zero for counters), rather than treating this as an error?
- If an object is in the "closed" state, can't we just return the last stats object before it was closed?

Statistics API Error Handling (cont'd)

Proposal:

- Delete the text in bullet 1.
- Add to Section 13.1: "If the object has not yet begun to send or receive data, the returned stats will reflect this. If the object is in the closed state, the returned stats will reflect the stats at the time the object transitioned to the closed state."

RTCDtmfSender sync with 1.0 (Issue 215)

Goal: same behavior (and spelling!) as in WebRTC 1.0.

But what is that?
11 June Editor's draft (http://w3c.github.io/webrtc-pc/#peer-to-peer-dtmf) Section 7:

Same spelling in WebRTC 1.0 and ORTC:
insertDTMF, onetonechange, toneBuffer, duration, interToneGap

Different spelling in WebRTC 1.0 and ORTC:
RTCDTMFSender vs. RTCDtmfSender in ORTC)

RTCDtmfSender sync with 1.0 (cont'd)

Peter Thatcher's suggestion on the public-webrtc mailing list:
https://lists.w3.org/Archives/Public/public-webrtc/2015Jun/0052.html

I like your rule, which I read as "we use CamelCase, except for that RTC thing at the beginning that we're stuck with".

I'm in favor of changing DTMFSender to be DtmfSender. I don't think there are any backwards compatibility issues with changing the type name (it's just a search and replace in the spec and code base). While we're at it, can we change the event objects with "RTCDTMF" to "RtcDtmf" as well? Might as well be consistent.

The only change that would have some compatibility implications would be the insertDTMF method. As much as I would like that to be insertDtmf, I'm willing to live with it being insertDTMF.

Questions for the CG

- Is the CG generally OK with the direction in which the Editor’s draft is headed?
- Do you have questions about general aspects of the spec?

For Discussion Today

- Issues from Phillip's Review (Issues 212, 213)
- DTLS
  - Certificates and Fingerprints in the non-mux case (Issue 210)
  - DtlsTransport.getLocalParameters and certificate creation (Issue 211)
  - DTLS: Problems with forking (Issue 218)
- Response to connectivity checks prior to calling iceTransport.start()? (Issue 170)

Coming Attractions

- Behavior of `sender.send(parameters)` and `receiver.receive(parameters)` when ***parameters.encodings*** not set.
- Statistics API (updates to support simulcast and scalable video coding)
- Updates for compatibility with WebRTC 1.0 objects
- IdP (if updated in WebRTC 1.0)
- Data Channel (if updated in WebRTC 1.0)

Support for “generation” (Issue 212)

From Philipp:

How does RTCIceCandidate does deal with the extensibility defined in RFC 5245 -- https://tools.ietf.org/html/rfc5245#section-15.1 (extension-att-name etc). For example, the generation attribute from http://xmpp.org/extensions/xep-0176.html#protocol-syntax is pretty common (e.g. supported in Chrome).

Robin Raymond:

`generation` does not apply to ORTC since the candidates come from the gatherer which never changes its usernameFragment / password. The restart mechanism happens in the IceTransport which can take a substitute gatherer. So the tracking of the `generation` has to happen at a high level.

Privacy (Issue 213)

From Philipp Hancke's review comments:
18) page 12, section 3.11 (RTCIceCandidate)

```
DOMString relatedAddress = "";

unsigned short relatedPort;
```

I don't think those attributes are useful, just a potential leak of ip addresses when forcing turn-only relays. So I would not expose them.

32) page 20, section 5.8

can be used to reduce leakage of IP addresses in certain use cases.

add a note about setting rel-addr to 0.0.0.0 then

Privacy (cont'd)

Martin Thomson:

Yes, this is a privacy issue, but those values are used to handle some corner cases in the deduplication algorithm. I'd be OK with them being replaced with (salted) hashes or something like that, but then you wouldn't be able to use SDP.

Robin Raymond:

I think the real answer is that the browser will have to enact a privacy mode to hide away the real IP addresses…

Question: Given the desire to maintain compatibility with WebRTC 1.0, is there anything we should do in the ORTC API?

DTLS Certificate Issues (Issues 210,211,218)

- Issue 210: In the RTP/RTCP non-mux case with DTLS, do we have distinct certificates for RTP and RTCP (and multiple fingerprints), or just one?
- Issue 211: When are DTLS certificates generated? When the DtlsTransport is constructed? When .getLocalParameters() is called? (needs a Promise), or some other time?
- Issue 218: How does forking work with DtlsTransport? For fingerprint verification, Answerers expect the DtlsTransport to provide a certificate matching the fingerprint provided in the Offer. This implies that the Offerer constructs DtlsTransports with the same certificate and fingerprint for each fork.

Are these three different problems (each with their own solution), or three symptoms of the same problem?

Three different problems?

- Issue 210: In the RTP/RTCP non-mux case with DTLS, do we have distinct certificates for RTP and RTCP (and multiple fingerprints), or just one?
  - ***Can generate distinct certs and fingerprints (interop issues?) OR***
  - ***Can mandate the same certificate/fingerprint.***
- Issue 211: When are DTLS certificates generated?
  - ***Can make .getLocalParameters a promise.***
- Issue 218: How does forking work with DtlsTransport?
  - ***Could have a cloneTransport method for DTLS that constructs an DtlsTransport with the same certificate/fingerprint (yuck!)***

Or three symptoms of the same problem?

- WebRTC 1.0 Section 5.5 Certificate management API (currently optional):

```
partial interface RTCPeerConnection {
    static Promise<RTCCertificate> generateCertificate (AlgorithmIdentifier keygenAlgorithm);
};
```

- Issue 210: RTP/RTCP non-mux case with DTLS. ***Add*** **RTCCertificate** ***as a required argument to the DtlsTransport constructor.***
- Issue 211: When are DTLS certificates generated? When generateCertificate ***is called.***
- Issue 218: How does forking work with DtlsTransport? ***Pass the same certificate to the DtlsTransport constructor for each fork.***

Proposed RTCDtlsTransport changes

```
partial interface RTCCertificate {
    static Promise<RTCCertificate> generateCertificate (AlgorithmIdentifier keygenAlgorithm);
};

[Constructor(RTCIceTransport transport, RTCCertificate certificate)]
partial interface RTCDtlsTransport {
    //...
};
```

Reasoning:

- Matches proposed pattern from WebRTC 1.0 (except we don't have a peer connection object)
- Certificate management API is now required but solves the issue(s)

Responding to connectivity checks (Issue 170) (IMPORTANCE - reminder)

Responding to incoming ICE connectivity checks immediately is desirable to provide faster media setup:

1) Avoid buffering incoming connectivity checks arriving to the ***RTCIceGatherer*** (to deliver to an ***RTCIceTransport*** later once `RTCIceTransport.start()` is called after the "answer" arrives).
2) Avoid setup delays by responding to incoming connectivity checks, allowing DTLS to be negotiated and setup earlier than full round trip signalling normally requires.
3) Correctly calculate the round trip time for connectivity checks (for better metrics).
4) Avoid loss of media due to buffer overflows:
    a) Incoming media can arrive once DTLS has entered the “connecting” state (after a successful connectivity check response is sent).
    b) If `receiver.receive()` is called prior to receipt of the Answer, for video, we can avoid having an initial I-frame overflow the buffer, which could result in multiple packet losses/frame loss despite robustness measures (RTX, FEC).

Responding to connectivity checks (Issue 170) (Agreed Solution "RTCIceTransport constructor")

```
[Constructor(RTCIceGatherer)]
partial interface RTCIceTransport {
  readonly attribute DOMString remoteUsernameFragment; // can get auto-filled
  readonly attribute RTCIceRole role;                  // can get auto-filled
  attribute EventHandler? onremotetransportlatched;    // needed? use case?
};

// Example:
var iceGatherer = new RTCIceGatherer(..);
var iceTransport = new RTCIceTransport(iceGatherer,..);
var dtlsTransport = new RTCDtlsTransport(iceTransport,..);
```

Responding to connectivity checks (Issue 170)
Proposed solution to race condition

Problem:

Potential race condition between programmer calling `RTCIceTransport.start()` with a usernameFragment / role different that an auto-latched incoming remote ICE usernameFragment / role due to asynchronous API;

Solution:

Add a factory method that constructs a brand new RTCIceTransport object with a specific username fragment or fetch a previously associated RTCIceTransport which was auto-latched to the usernameFragment passed in an atomic fashion.

```
[Constructor(optional RTCIceGatherer gatherer)]
partial interface RTCIceTransport {
    //...
    static RTCIceTransport createOrUseExisting(RTCIceGatherer gatherer, DOMString usernameFragment);
};
```

Responding to connectivity checks (cont'd)
(Race condition for incoming DTLS packet before DTLS constructed)

Problem:

Very small window where ***RTCIceTransport*** constructed from an ***RTCIceGatherer*** (and ready to respond to connectivity checks), but ***RTCDtlsTransport*** has not yet been constructed from the ***RTCIceTransport*** (so buffering is still needed):

```
var iceGatherer = new RTCIceGatherer(..);
var iceTransport = new RTCIceTransport(iceGatherer,..);
// tiny race window where ICE response can be sent but no DTLS transport wired yet
var dtlsTransport = new RTCDtlsTransport(iceTransport,..);
```

Solution:

Buffer a few packets to prevent the DTLS "hello" packet being lost / retransmitted.

Responding to connectivity checks (Issue 170) (Questions for ORTC CG)

1) Are these solutions "acceptable"?
2) Are the solutions worth the effort?

   NOTE: Many ICE scenarios cannot be set up until full round trip signalling happens anyway due to firewall pinholes not being opened until outgoing checks are issued.

3) Any other options?

Question for Community Group:

Are there any comments regarding known deficiencies of the ORTC API at this time?

Good time to speak before implementation is too far along!

Organization / Call for implementation feedback

Mobile C++ ORTC implementation:
https://github.com/openpeer/ortc-lib-sdk

ORTC JS "shims" (i.e. downshim and upshim to / from WebRTC 1.0)
https://github.com/openpeer/ortc-js-shim (vacant repo)

ORTC specification:
https://github.com/openpeer/ortc

ORTC Node JS implementations:
https://github.com/openpeer/ortc-node

Browser Implementations:
Requested at this time (status.modern.ie lists ORTC as "**In Development**")

ORTC-lib Update

Work on ORTC-lib continues.

If you would like to participate in coding any of the ORTC open source projects then:

1. Join ORTC Community Group
2. Join developer mailing list / group
   http://ortc.org/dev
3. Start helping!

Thank you

Special thanks to:

Bernard Aboba - Microsoft

Michael Champion - Microsoft

Justin Uberti - Google

Peter Thatcher - Google

Robin Raymond - Hookflash

Erik Lagerway - Hookflash

For More Information

ORTC Community Group

http://www.w3.org/community/ortc/

ORTC Developers & API Drafts

http://ortc.org