WebRTC states and callbacks

This document suggests the important states that we want to capture for signaling and ICE negotiation, which are referred to as PeerState and IceState in the PeerConnection spec.

PeerState proposal

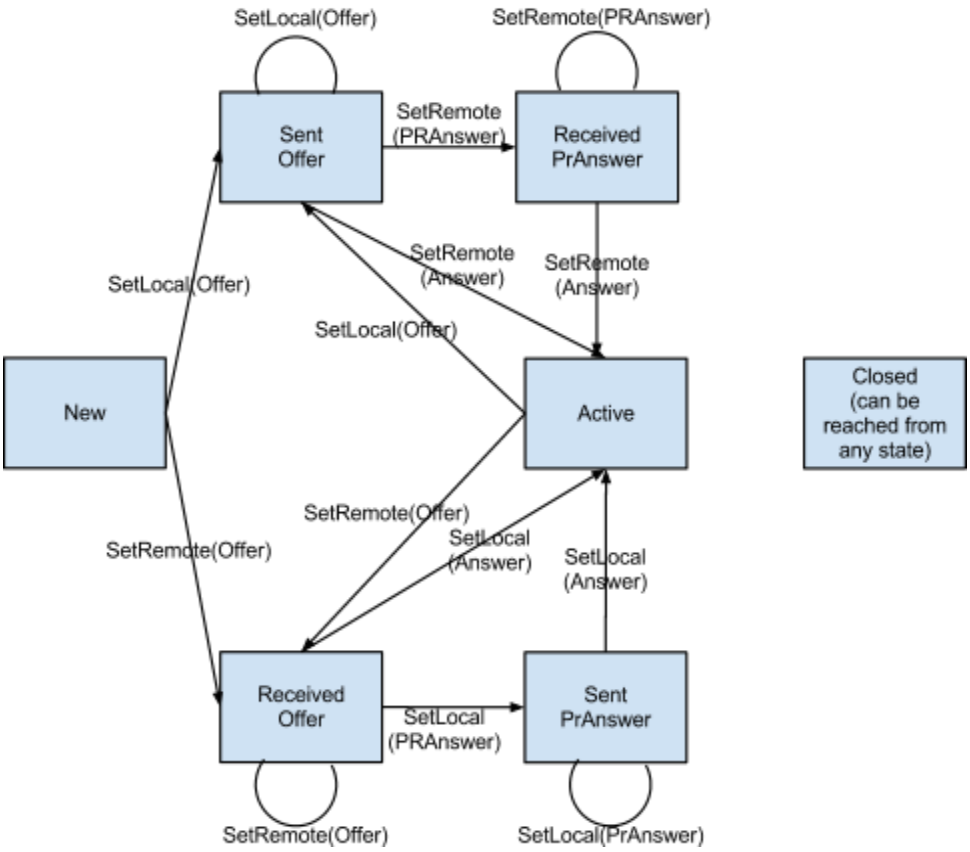
```
enum PeerState {
  "new",
  "sent-offer",
  "received-offer",
  "sent-pranswer",
  "received-pranswer",
  "active",
  "closed"
}
```

State descriptions

"new"
The object was just created, and no descriptions have been set.
"sent-offer"
A local description, of type "offer", has been supplied.
"received-offer"
A remote description, of type "offer", has been supplied.
"sent-pranswer"
A remote description of type "offer" has been supplied and a local description of type "pranswer" has been supplied.
"received-pranswer"
A local description of type "offer" has been supplied and a remote description of type "pranswer" has been supplied.
"active" (also could be called "open", "stable")
Both local and remote descriptions have been supplied, and the offer-answer exchange is complete.
"closed"
The connection is closed.

Note: "opening" and "closing" have been removed, as they don't reflect any real-world state of the PeerConnection.

State diagram



Example

Caller transition:

- new PeerConnection(): new
- setLocal(offer): sent-offer
- setRemote(pranswer): received-pranswer
- setRemote(answer): active
- close(): closed

Callee transition:

- new PeerConnection(): new
- setRemote(offer): received-offer
- setLocal(pranswer): sent-pranswer
- setLocal(answer): active
- close(): closed

IceState proposal

```
enum IceState {
  "starting",
  "checking",
  "connected",
  "completed",
  "failed",
  "disconnected",
  "closed"
}
```

State descriptions

"starting"
The ICE Agent is gathering addresses and/or waiting for remote candidates to be supplied.
"checking"
The ICE Agent has received remote candidates on at least one component, and is checking candidate pairs but has not yet found a connection. In addition to checking, it may also still be gathering.
"connected"
The ICE Agent has found a usable connection for all components but is still checking other candidate pairs to see if there is a better connection. It may also still be gathering.
"completed"
The ICE Agent has finished gathering and checking and found a connection for all components.
"failed"
The ICE Agent is finished checking all candidate pairs and failed to find a connection for at least one component.
"disconnected"
Liveness checks have failed for one or more components. This is more aggressive than "failed", and may trigger intermittently (and resolve itself without action) on a flaky network.
"closed"
The ICE Agent has shut down and is no longer responding to STUN requests.

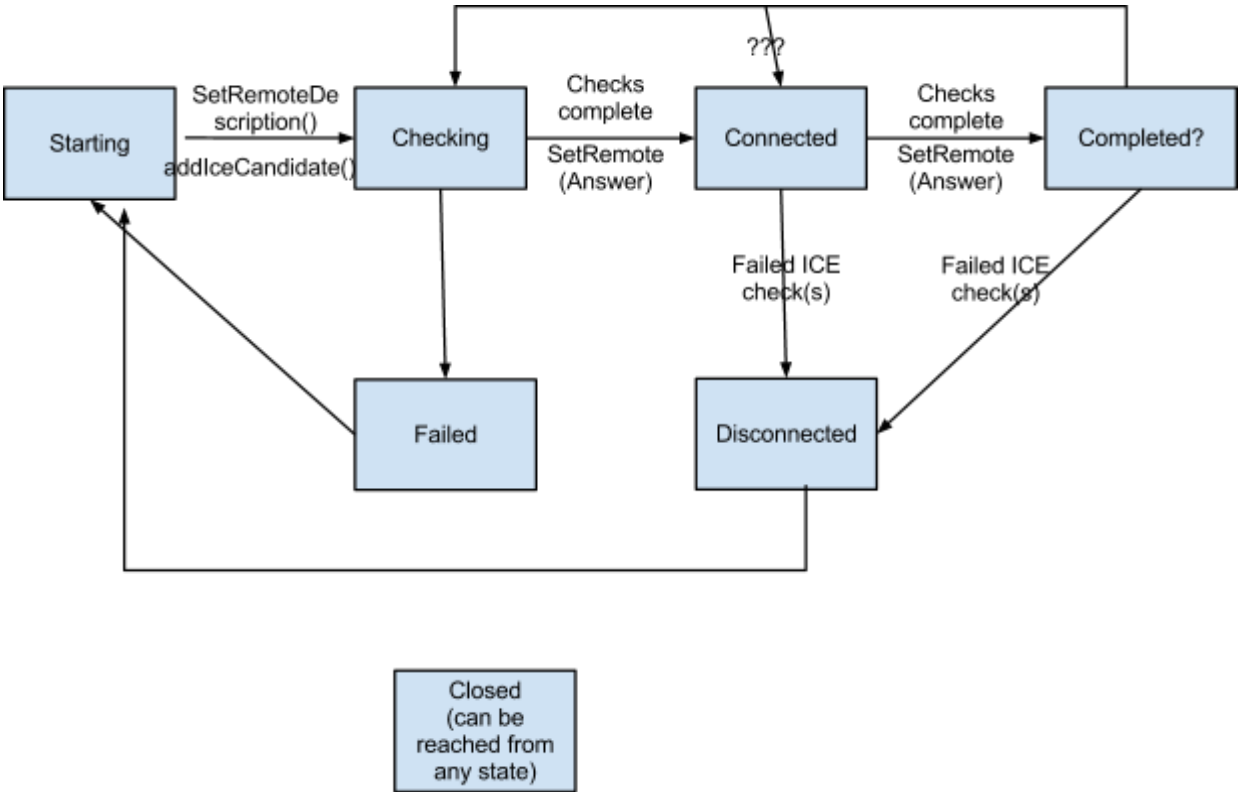
Note that"new" has been removed, as such a state no longer exists; "waiting"/"gathering" have been merged into "starting", as gathering is somewhat orthogonal to the checking/connected/completed states, and is now handled via the onicecandidate callback (where a NULL candidate means that gathering has completed.)

Note also that the states take either the value of any component or all components, as outlined below:

- "checking" occurs if ANY component has received a candidate and can start checking
- "connected" occurs if ALL components have established a working connection
- "completed" occurs if ALL components have finalized the running of their ICE process
- "failed" occurs if ANY component has given up trying to connect
- "disconnected" occurs if ANY component has failed liveness checks
- "closed" occurs only if PeerConnection.close() has been called.

If a component is discarded as a result of signaling (e.g. RTCP mux or BUNDLE), the state may advance directly from "checking" to "completed".

State diagram



Example

Caller/callee transition:

- new PeerConnection(): Starting
- (Starting, remote candidates received): Checking
- (Checking, found usable connection): Connected
- (Checking, gave up): Failed
- (Connected, finished all checks): Completed
- (Completed, lost connectivity): Disconnected
- (any state, ICE restart occurs): Starting
- close(): Closed

PeerConnection callbacks proposal

Remove onopen, onconnecting - redundant with other callbacks. Other callbacks defined as below:

```
// Fires whenever readyState changes, either from setLocalDescription,
// setRemoteDescription, or close. Does not fire for the initial state.
attribute Function?         onstatechange;
// Fires whenever iceState changes [needs review of ice states].
// Does not fire for the initial state.
attribute Function?         onicechange;
// Fires whenever a new local ICE candidate is available
```

attribute Function? onicecandidate;
// Fires whenever a new stream appears in a remote description.
// This will be fired only as a result of setRemoteDescription.
attribute Function? onaddstream;
// Fires whenever a stream is removed from a remote description.
// This will be fired only as a result of setRemoteDescription.
attribute Function? onremovestream;
// TBD
attribute Function? ondatachannel;
// TBD
attribute Function? onrenegotationneeded;

onaddstream happens as early as possible and before any the “SDP rejects/ or accepts” a given thing. When the SDP accepts something, you get the addTrack callback. Later if SDP ended a media flow, that would result in trackEnded callback.