

ICE States

W3C WebRTC
September 2012
Cullen Jennings

Review of ICE State in RFC 5245

- Controlling / non controlling
- Check List: Running, Failed, Completed
- For each candidate:
 - State: Waiting, In Progress, Succeeded, Failed, Frozen
 - Nominated or not Nominated
- 5 states for each candidate, lots of candidates

Things to keep in mind while designing this

- In some cases such as no multiplexing, there are a bunch of ICE machines running at the same time in the PeerConnection
- For some apps, need to know when gathering is done so you know offer will not get more candidates
- For some apps, need to know when checking is done so you know path will not changes. (Changing the path changes latency, congestion, etc)
- After everything is connected and stable, new interfaces can be added
- UX need rapid notification of when things get connected or fail

ICE State Summarization for Single Flow

- At pretty much any time you can be gathering or not gathering and simultaneously checking or not checking and may or may not have connectivity across the flow
- Leads to states of:
 - Idle: not started yet, waiting for stun/turn servers etc
 - Gathering
 - Waiting: gathering has completed but no remote to start checking
 - Checking + Gathering
 - Checking
 - Checking + Gathering + Connected
 - Checking + Connected
 - Connected: found at least one path to remote side
 - Failed: all candidates have been checked and no path found
 - Closed
- Important events for the JS are:
 - Done gathering, (re)started gathering
 - Got connected, No longer connected
 - Done checking, (re)started checking

Multiple ICE Machines

	Idle	Gathering	Checking + Gathering	Waiting	Checking	Check+Gath + Connected	Checking + Connected	Connected	Failed	Closed
Idle	Idle									
Gathering	x	gathering								
Checking + Gathering	x	gathering	gathering							
Waiting	x	gathering	gathering	waiting						
Checking	x	gathering	gathering	checking	checking					
Check+Gath + Connected	x	partially connected	partially connected	partially connected	partially connected	connected				
Checking + Connected	x	partially connected	partially connected	partially connected	partially connected	connected	connected			
Connected	x	partially connected	partially connected	partially connected	partially connected	connected	connected	connected		
Failed	x	partially failed	partially failed	partially failed	partially failed	partially failed	partially failed	partially failed	failed	
Closed	x	x	x	x	x	x	x	x	x	Closed

What to do?

Proposal A

- Have an enum with states from previous table:
 - Idle, gathering, waiting, checking, partially connected, connected, partially failed, failed, closed
- Event any time the state changes
- Somewhere deep in the statistics interface be able to get the exact state of each ICE machine for detailed debugging

Issues with Proposal A

- No event when all gathering is done if one ice machines gets connected before the other ice machines finish gathering - probably not a big deal
- If there are three ice machines, no event when 2nd one becomes connected
- Is it more important to notify of failure or connected?
- No event when all paths have stopped changing (could fix by adding an allConnected state)
- Code ends up with lots of logical combinations of states
 - if connected or partially connected

Proposal B

- Have a few functions that return if all, some, or none of the ICE state machines are doing the following:
 - isGathering()
 - isChecking()
 - isConnected()
 - isFailed()
 - and a boolean isClosed()
- Have events for
 - gathering has (re) started, all gathering has completed
 - checking has (re) started, all checking is completed
 - any ICE machine becomes connected, all ICE machines are connected
 - any ICE machine goes to failed
- Somewhere deep in the statistics interface be able to get the exact state of each ICE machine for detailed debugging

Issues with Proposal B

- Not as much experience writing code with this API. Might be ugly

Proposal C

- Create a transport object associated for each Track
 - Can't use Track object directly as single track and be in more than one Stream
- Each Transport would report state for just a single ICE machine
- States would be:
 - Idle
 - Gathering
 - Checking + Gathering
 - Waiting
 - Checking
 - Check+Gather+Connected
 - Checking+Connected
 - Failed
 - Closed

Issues with Proposal C

- Passes on the problem of merging states to the application
 - This could be pro or con
- pros/cons to introducing an Transport object