Trickle ICE

IETF 95, Buenos Aires

Justin Uberti
Peter St. Andre

Issue #1: Handling Duplicate Candidates

Problem: A trickled candidate already exists, but with a different priority
(for example: prflx then srflx)

Option A: First one wins (ignore the duplicate)

Option B: Last one wins

Option C: Update the priority to highest (like 5245: "The agent SHOULD eliminate the redundant candidate with the lower priority:)

Recommendation: C

By the way, this is also an issue with 5245, not just trickle. So maybe we should put it in 5245bis.

Issue #2: Freezing Behavior

Problem: Candidates A and B share a foundation. A is "first", so B is frozen. But A never comes, so B stays frozen.

Option A: Unfreeze B because it arrived before A (and also unfreeze A if it arrives)
Problem: Might have some "extra" unfrozen candidates.

Option B: Don't do freezing.
Problem: Unbundled apps *might* have slower connection time.

Option C: Only trickle in order. Unfreeze B because it's first.
Problem: Delays trickling. Can't be enforced by WebRTC apps.

Recommendation: Option A

Issue #3: Trickling After ICE Restart

Problem

An ICE restart happens before candidate gathering is complete. How do you trickle those candidates?

Option A

Signal a way to tell whether the candidate is from before or after the ICE restart. For example, signal the ufrag.

Option B

Don't trickle candidates unless they apply to the latest ufrag/pwd.
Problem 1: The ICE restart can be rejected (or possibly rolled back in WebRTC)
Problem 2: A suboptimal connection until the restart completes.

Recommendation

Option A. This makes it absolutely clear which candidates go with which ICE gather.