Glare Handling in WebRTC Signalling

Magnus Westerlund

draft-jennings-rtcweb-signaling-01

Outline

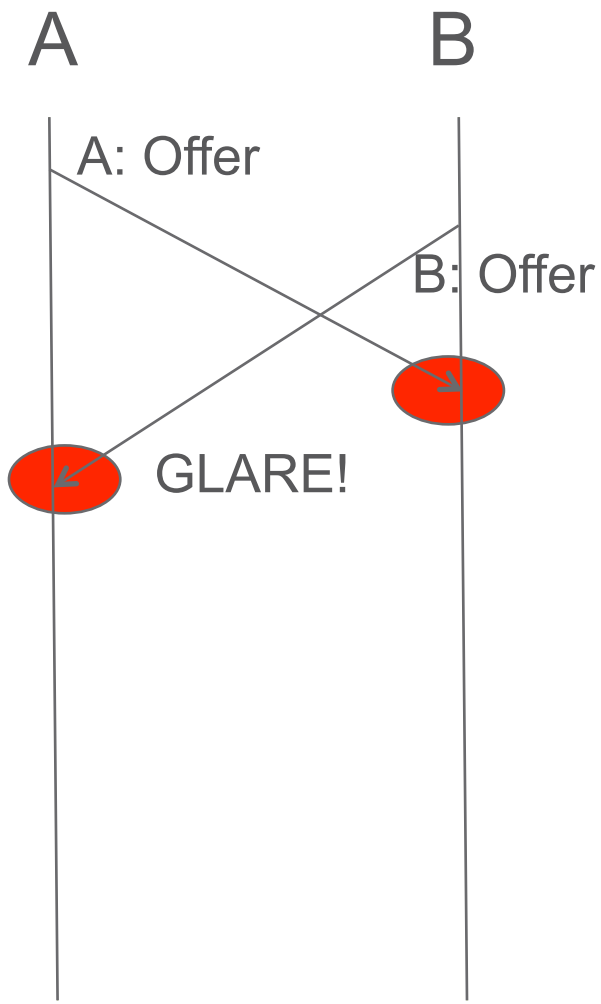
- What is Glare
- Solution Proposal
- Interworking Support

What is Glare

- Glare is when the signalling is not coordinated, simultaneous and risks ending up in an undetermined state
- In the context of ROAP and WebRTC the situation that occurs are:
  - Both end-points make an ROAP offer at the same time
- When can this occur
  - Initially
  - When updating an established session

Glare in Established Session

- A and B's Offers indicate the same ROAP session
- In this case where only end-points check for GLARE both A and B will see the other side's ROAP message
- A will detect Glare when B's Offer arrive and A has an outstanding Offer
- Same thing for B.
- So both end-points end up in state which needs resolution.



Handling Initially Glare

- Two WebRTC clients running the same Web application initiates a session to each other
  - The Web application will need to determine if these are for the same purpose
  - If they are for the same purpose, ROAP glare handling could be used.
  - If not the application could open several sessions or determine which purpose that is the prioritized.
- ROAP message could deal with initial glare assuming the application determines that the offers are for the some purpose
  - As much a API question as protocol one
  - Alternatively the application could simply use the ROAP mechanism values at the application level

Glare Resolution Proposal

- All ROAP Offers include a tiebreaker:
  - 32-bit unsigned integer with a random number.
  - 0 and 0xffffffff are special purpose.
- When a glare occur the two Offers tiebreakers are compared
  - The Offer with the larger tiebreaker wins and goes ahead
  - The Offer that loses are immediately discarded
  - If they values are equal an error is sent and both end-point may retry with new random values
- The special values are used by gateways for better support when interworking with other protocols

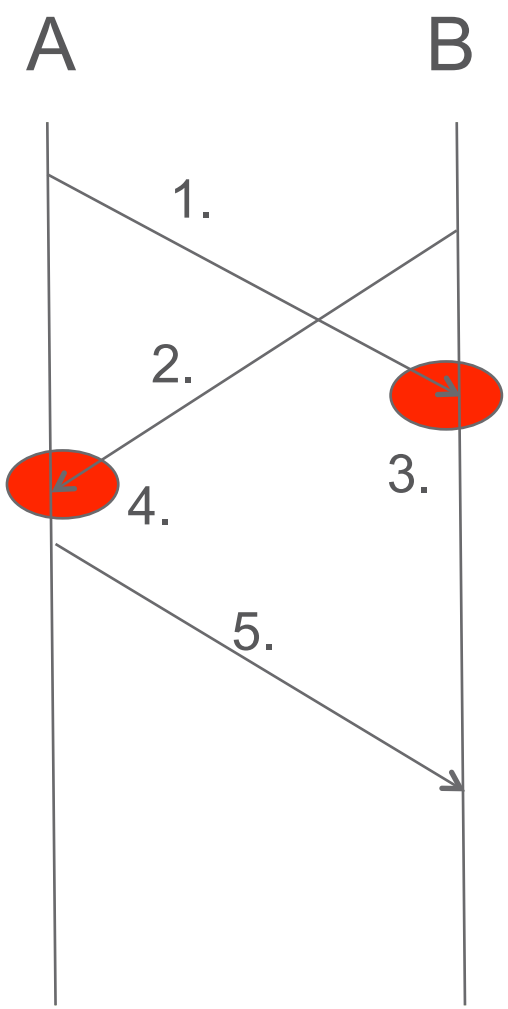
Why not use the SIP mechanism

- The SIP mechanism is based on roles and timers
  - The owner of the call-id in the SIP dialog backs of 2.1 - 4.0 s
  - The other backs off 0 - 2.0 s
  - When timer expires re-try the offer
- Can't be used to deal with initial glare (handled by SIP)
- WebRTC signalling message transport may be slow:
  - Client to server short poll 1s or more just in waiting + transport delay
  - The difference in the back-off might be completely disappear
    - Causing a new glare
- Sub-optimal in that it can't determine in most cases which Offers goes ahead immediately

Example

- Clients A and B has an established session
- All the below is happening within that context:

1. A sends an Offer to B with tiebreaker value of 25
2. B sends an Offer to A with tiebreaker value of 40
3. B detects the glare condition when A's message arrive. B determines it's Offer won and discards A's
4. A detects the glare condition when B's message arrive. A determine it's Offer lost and process B's Offer
5. A send an Answer to B's Offer



Early Processing

- As the process is deterministic any intermediary can also run the process as long as it knows the previous offer will reach the end-point
  - If it detects a glare it can possibly suppress the losing message
  - Assumes reliable message transport

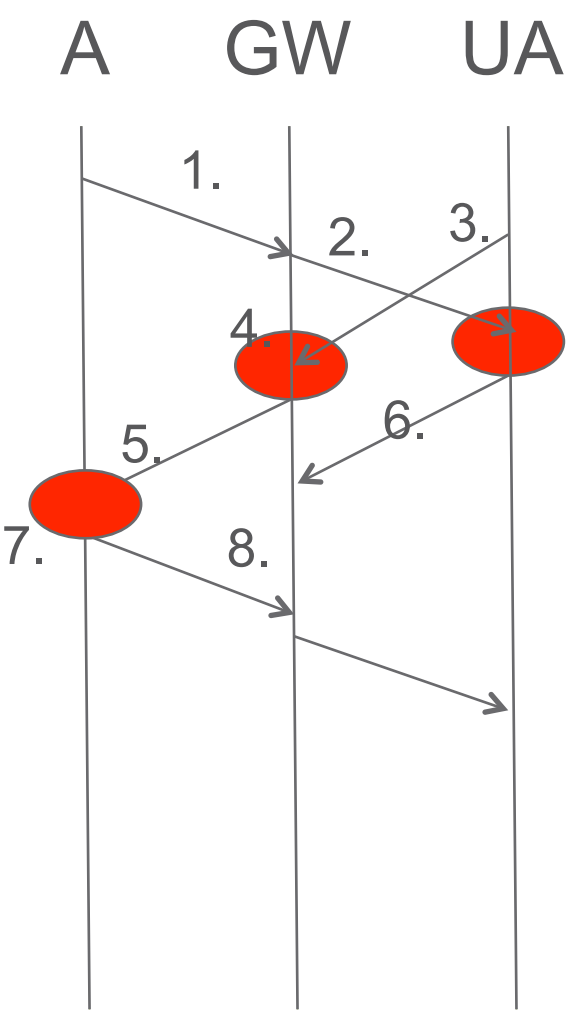
Interworking

- The Glare mechanism has been designed with Interworking in consideration
- The special tiebreaker values 0 and 0xffffffff are used by gateways to allow them to win or loose on purpose to minimize issues in the other protocol

Interworking Example 1

- WebRTC client A uses Gateway (GW) to establish a session with SIP User Agent (UA)
  - GW is owner of the Call-ID in the SIP Dialog

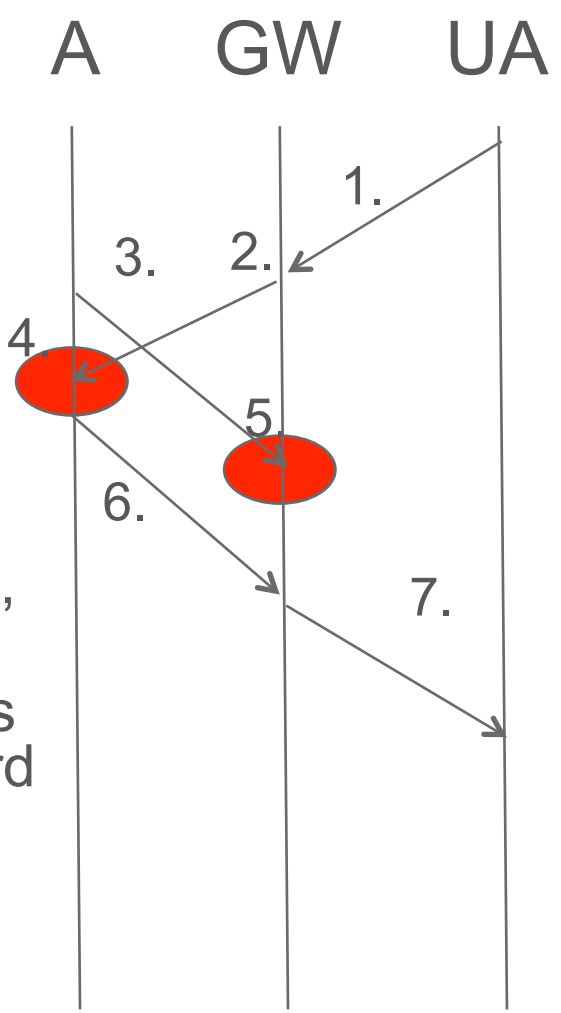
1. A sends an Offer
2. GW forwards this as an SIP Update
3. UA sends a SIP Update
4. GW detects Glare have happened
5. GW sends an Offer with the Tiebreaker value set to 0xffffffff
6. UA sends a 491 which GW accepts and take no further action on
7. A detects Glare, determines it lost, process the Offer
8. A sends the answer to the GW that generates the SIP ACK



Interworking Example 2

- WebRTC client A uses Gateway (GW) to establish a session with SIP User Agent (UA)
  - GW is owner of the Call-ID in the SIP Dialog

1. UA sends a SIP Update targeted to A to the GW
2. The GW creates a ROAP Offer with a Tiebreaker value of 0xffffffff
3. A sends an Offer to the GW with a random tiebreaker
4. A detects Glare and determines its Offer lost, process the offer from the GW.
5. Meanwhile the GW will detect the glare on its side. Determine that its Offer won and discard A's
6. A sends an answer to the GW
7. GW translates this into a SIP ACK



Open Questions

- Does anyone see a need to send an error for the Offer that looses?