Network Working Group
Internet-Draft
Updates: 3264 (if approved)
Intended status: Standards Track
Expires: April 20, 2014

A. B. Roach
Mozilla
S. Nandakumar
Cisco

October 17, 2013

Using Partial Offers and Partial Answers in a Multimedia Session
draft-roach-mmusic-pof-pan-01

Abstract

Whenever two hosts have the ability to set up and control a session on a peer-to-peer basis, situations can arise in which both parties attempt to change session parameters "at the same time," such that the session control messages cross on the wire. When this happens, implementations need to invoke extraordinary procedures to return the shared state of the session to a common view between the endpoints.

For real-time communications, these session control messages are typically exchanged using the session description protocol (SDP), using an Offer/Answer model. This document expands the offer/answer model to include the ability to exchange information relating to discrete media streams within the session. By reducing the amount of session data, the frequency of session state conflicts can be reduced; and, for certain types of operations, conflicts can be eliminated altogether.

This document updates RFC 3264.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 20, 2014.

Table of Contents

1. Introduction

The SDP [RFC4566] offer/answer model defined in [RFC3264] briefly mentions "glare" as a potential issue in the use of offer/answer exchanges, although it relegates the problem to the "higher layer protocol" to resolve. In SIP [RFC3261], resolving state after a glare condition is performed via a timer-based back-off mechanism. For WebRTC, detection of glare comes in the form of an "InvalidStateError" exception. Actual resolution of glare is currently undefined; the present assumption is that the applications that make use of RTCWEB are responsible for handling glare in a sensible fashion.

The penalty for glare isn't simply code complexity; it results in delays in updating sessions state, which can end up visible to users, leading to a less optimal user experience.

Many of the emerging uses for both SIP and RTCWEB involve sessions with a large number of media streams, with streams being added and removed frequently. This kind of session churn increases the incidence of glare significantly.

To reduce the incidence of glare under these circumstances, this document defines a procedure via which partial offer/answer exchanges may take place. These exchanges operate on one or more media sections at a time, rather than an entire SDP body. These operations are defined in a way that can completely avoid glare for stream additions and removals, and which reduces the chance of glare for changes to active streams. This approach requires all media sections to contain an "a=mid" [RFC5888] attribute.

This document focuses on the application of this technique for use in RTCWEB and WebRTC. The author anticipates that future work will describe its use in conjunction with SIP and SIP-derived technologies (such as multiparty conferencing and telepresence).

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Mechanism Overview

The core of this mechanism is the concept of "partial offers" and "partial answers." Syntactically, these entities are SDP fragments, consisting of exactly one o= line; one or more media sections; and any i=, c=, b=, k=, and a= lines associated with the media sections. They are formatted exactly as they would be if they were part of a larger SDP document, with one key exception: unlike SDP, in an SDP fragment, the ordering of media sections relative to each other is not significant. Note that SDP fragments contain only that information that pertains to media. Other than the mandatory o= line, they never contain any session-level information. Within the o= line, only the <sess-version> field is allowed to be changed from its previous value. Any changes to session-level information are expected to use a full offer/answer exchange rather than the partial offer/partial answer mechanism defined by this document.

> OPEN ISSUE: Do we need to relax this session-level prohibition? It makes the mechanism clean. However, it does make it difficult to add a new stream while simultaneously associating it with a group. On the other hand, group lines don't have unique identifiers, so just sending a single group line over can be ambiguous. One way around this would be requiring that partial offers and partial answers must contain all group attributes associated with the session, but this still gets potentially messy if we have both sides trying to update group information at the same time. An alternative approach might be to indicate group information only for *new* streams, and require full offers for any other group changes. Of course, without unique group identifiers, we're still stuck with the challenge of unambiguously identifying which group we're adding a new line to. If we constrain group membership so that each group can be uniquely identified by its type and members, then that should be sufficient. Is such a constraint acceptable?

Using this mechanism has two key prerequisites: (1) all offer/answer exchanges in the session prior to sending a partial offer have contained "a=mid" attributes for each media section, and (2) both sides are known to support the partial offer/answer technique (either because they are part of a single domain of control, or because use of this technique has been explicitly signaled).

The use of an SDP fragment body will be explicitly signaled, e.g., using a different MIME type for SIP, or using a different "type" field for the WebRTC API.

3.1. Adding a Stream

To add a stream glarelessly, a party creates a "partial offer" consisting of an o= line and one or more media sections, including all of the corresponding i=, c=, b=, k=, and a= lines. Each media section contains an "a=mid" attribute, indicating an MID that has not yet been used in the session.

Upon receipt of a partial offer, an implementation processes each media section independently. For each media section, the recipient examines the MID in it. If the MID does not match any existing MID in the session, then it represents a new media stream. Assuming the recipient does not have an outstanding, unanswered partial offer that also adds a stream, this new media section is simply appended to the end of the existing session description, the SDP sess-version is increased, and an answering media section is created. Once all answering media sections have been processed, they are concatenated into a partial answer. This partial answer consists of one or more media sections, each containing an MID matching the one from the partial offer.

If the recipient of a partial offer that contains a new MID has also sent a partial offer adding a new stream to the session, then ambiguity can arise regarding the canonical ordering of media sections within the session description. In this situation, both partial offer/answer exchanges are allowed to complete independently (as no fundamental data glare has occurred). However, the order in which they are appended to the session description is synchronized by performing a lexical comparison among each media section’s MID attribute: the media sections are appended to the session in lexically increasing order.

3.2. Changing a Stream

Partial offers may also be generated for modification of an existing stream. In this case, the MID in the media section of a partial offer will match an existing MID in the session description.

Upon receipt of a partial offer, an implementation examines the MID in it. If the MID matches any existing MID in the session, then it represents a modification to that media section. Assuming the recipient does not have an outstanding, unanswered partial offer that also modifies that exact same stream, this media section is treated as an independent renegotiation of that stream (only). The SDP

version is increased, and a partial answer is created. This partial answer consists of an media section and its attributes, and has an MID matching the one from the partial offer.

> OPEN ISSUE: Since stream *changes* can result in glare, the foregoing text assumes that only one media section will be sent for such a change. Is this okay?

If the recipient of a partial offer that contains an existing MID has also sent a partial offer to change that exact same stream, and neither the received nor the sent partial offer contains an "a=inactive" attribute, then a legitimate glare condition has arisen. Normal glare recovery procedures -- e.g., using a tie-breaker token or a back-off timer -- must be engaged to resolve the conflict.

3.3. Removing a Stream

To remove one or more a streams in a way that eliminates the chance of glare, an implementation generates a new partial offer, containing one or more media sections. Each media section contains an MID matching the stream it wants to remove, and indicates a transport port of zero, indicating that the stream is being deactivated.

If the recipient of a partial offer that contains an existing MID has also sent a partial offer to change that exact same stream, and either one of the received or the sent partial offer contains a port number of zero, then the stream is deactivated. At this point, both partial offers are discarded, the corresponding media section in the session is modified by changing its port to zero, and a partial answer is generated representing this single change.

4. Use With Other Protocols

Note that this document simply defines the extensions to the SDP offer/answer model for dealing with partial offers and partial answers. In the same way that [RFC3264] does not define specific SIP, JSEP, or WebRTC handling, neither does this document. In order for this technique to be useful, protocol-specific mechanisms need to be defined. This additional work is left to appropriate venues, such as the W3C WebRTC WG, the RTCWEB WG, and the SIPCORE WG. If the higher-level protocol allows the use of unordered message delivery, it is that protocol’s responsibility to ensure that the result of partial offer/partial answer exchanges is a shared and identical session state between the parties involved.

To assist in understanding the mechanism being proposed, we describe, in a very high-level and non-normative way, how this mechanism might be applied to a couple of specific higher-level signaling systems.

4.1. High-Level Sketch: Use With JSEP/WebRTC

For WebRTC, we envision that such additional specification would add a new constraint to createOffer, requesting that a partial offer be generated (if possible). The resulting RTCSessionDescription would contain only the media sections that have changed since the most recent offer/answer exchange, and would have a type of "partialOffer." When createAnswer is called after receipt of a partialOffer, it would create a partialAnswer, containing only the media sections referenced in the partial offer, that can be provided to the remote party.

4.2. High-Level Sketch: Use With SIP

For SIP, partial offers and partial answers will likely be provided in SIP UPDATE [RFC3311] or INFO [RFC6086] messages, containing a special "application/sdpfrag" MIME type [I-D.ivov-dispatch-sdpfrag], and a content-disposition that indicates that the contents are a partial offer (rather than, say, a trickle ice candidate). Although INVITE may seem like a natural fit for this kind of behavior, its current definition includes strong glare resolution behaviors that makes it unsuitable for this purpose. Naturally, any such mechanism will be paired with a SIP feature tag that allows for negotiation of support for partial offers and answers.

5. Protocol Operation

The following sections formally defines the procedures for generating and processing partial offers and partial answers.

At any time during an ongoing session, either agent in the session MAY generate a new partial offer that updates the session, subject to the restrictions described in the following sections. However, it MUST NOT generate a new partial offer if it has received any partial or full offer which it has not yet answered or rejected.

An agent also MUST NOT generate a partial offer if it has sent a partial or full offer which has not yet been accepted or rejected.

> OPEN ISSUE: It seems like we might be able to have multiple outstanding sent partial offers at once, as long as they don’t try to act on the same media section. The reason it’s disallowed in the above paragraph is that having several partial offers potentially outstanding in both directions makes it very, very, very complicated to resolve the ordering of media sections if these partial offers in opposite directions overlap temporally.

In the situations described as "glare" below, the higher layer protocol needs to provide a means for resolving such conditions. This will generally be the same mechanism used to resolve the glare conditions described in [RFC3264].

5.1. Common Procedures

For all of the procedures described in the following sections, whenever an o= line is included in a partial offer or partial answer, its <username>, <sess-id>, <nettype>, <addrtype>, and <unicast-address> values MUST be identical to those sent in the most recent full offer or full answer generated by this agent for this session. The <sess-version> value MUST be larger than the value in all previously sent offers, partial offers, answers, and partial offers generated by this agent for this session.

Whenever the procedures in the following sections indicate that a media section is to be included in a partial offer or partial answer, that media section MUST consist of an m= line along with all i=, c=, b=, k=, and a= lines associated with that media section. If a line is absent from a media section in a partial offer or partial answer, it MUST be interpreted as an explicit removal of that value from the media section. Recipients of such messages MUST NOT assume that a previously-established but omitted value is still in effect.

5.2. Generating a Partial Offer

Whenever an agent wishes to change the state of the media in an ongoing session -- whether through addition, modification, or removal of a stream -- it does so through either an offer or a partial offer. In deciding which to use, the implementation first verifies that it has received positive confirmation that the remote implementation supports the partial offer/partial answer mechanism. The means of negotiating such support is left to the higher-level protocol that makes use of the offer/answer model. The implementation then verifies that all media sessions in the current session are associated with unique MID values. Finally, the implementation evaluates whether the changes it needs to make can be performed exclusively using the values present in a media section, without any modifications necessary to session-level values (except for the sess-version value on the session-level o= line).

If all three of the criteria described above are true, then the implementation MAY send a partial offer to make the changes it wants to request. If any of these criteria are not true, then the implementation MUST use a full offer, according to the procedures described in [RFC3264].

Once the agent determines that the change it wishes to make is eligible to use the partial offer mechanism, it forms a new SDP fragment by following these steps:

1. For each desired change to an existing media section, the agent creates a new partial offer consisting of one o= line and a single media section. This media section MUST contain an "a=mid" attribute containing an MID that matches the media section that is being modified. The media section also contains the modifications that the agent wishes to make, as described in section 8.3 of [RFC3264]. This partial offer is sent in isolation, with no other media section changes, additions, or removals in the same partial offer.

2. If the desired change involves one or more media section additions or removals, the agent creates a new partial offer consisting of one o=line and any media section described in the following two steps.

3. For each new media section to be added, the agent creates a new media section to be added to the aforementioned partial offer. This media section MUST contain an "a=mid" attribute, and the MID present in this attribute MUST contain at least 32 characters chosen randomly from full set of 79 characters allowed in a token. The remainder of the media section contains the various values that the agent wishes to have associated with the corresponding media, and is created according to the procedures described in section 5.1 or 5.2 of [RFC3264], as appropriate.

4. For each existing media section to be removed, the agent creates a new media section to be added to the aforementioned partial offer. This media section MUST contain an "a=mid" attribute containing an MID that matches the media section that is being removed, and MUST contain a <port> value of 0 (zero). Except for the required "a=mid" attribute, this media section MAY omit any or all i=, c=, b=, k=, and a= lines, and MAY list only one m= line <fmt> value.

Once the preceding steps have been followed to create one or more partial offers, the agent makes use of the high-level signaling protocol to convey the offers to the remote agent, one at a time.

5.3. Processing a Partial Offer

Upon receipt of a partial offer, an agent first determines whether it has sent any full offers for the corresponding session. If it has, then the partial offer represents a glare condition that is resolved via the higher-level protocol. It then verifies whether it has

received any partial or full offers to which it has not yet sent an answer or a rejection. If so, then it rejects the partial offer as invalid behavior.

The agent then examines the o= line in the received partial offer. If If the <sess-version> value is less than the most recently received full (non-partial) offer or answer, then the partial offer is stale and MUST be rejected. The means for rejecting the partial offer are left to the higher-level protocol.

After such validation takes place, the agent iterates through each media section and performs the following steps:

1. If the MID present in the received media section matches a media section already present in the ongoing session and has a non-zero port number, it represents a change to an existing media stream.

   * If the partial offer contains more than one media section, then the recipient MUST reject the partial offer as invalid behavior.

   * If the MID matches the MID of a media section in a partial offer that the agent has sent, AND the sent media section contains a port number of zero, then the incoming partial offer is rejected, as any such changes have been "overtaken by events:" the stream will be deactivated momentarily.

   * The recipient verifies that the MID does not match the MID of any media section in any partial offers that it has sent but has not yet received a partial answer or rejection for, unless the media section in the sent partial offer has a port number of zero. If this verification fails, then the received partial offer represents a glare condition that is resolved via the higher-level protocol.

   * After the preceding verifications have succeeded, the agent creates media section to include in the partial answer. To reject the media section in the partial offer, the agent generates a media section with a port number set to zero; otherwise, the agent forms the media section by following the procedures described in section 6.1 or 6.2 of [RFC3264], as appropriate.

2. If the MID present in the received media section matches a media section already present in the ongoing session and has a port number of zero, then it represents the removal of an existing media stream. The agent creates a media section to include in the partial answer. With the exception of the "a=mid" attribute,

this media section MAY omit any or all i=, c=, b=, k=, and a= lines, and MAY indicate a single payload type.

3. If the MID present in the received media section does not match any media section already present in the ongoing session, then it represents a new media stream.

   * If the received media section contains a port number of zero, then the recipient MUST reject the partial offer as invalid behavior: this mechanism does not support the atomic addition and removal of the same stream.

   * If the above validation succeeds, the agent creates a media section to include in the partial answer. To reject the media section in the partial offer, the agent generates a media section with a port number set to zero; otherwise, the agent forms the media section by following the procedures described in section 6.1 or 6.2 of [RFC3264], as appropriate.

All media sections that are formed in the foregoing steps MUST contain an "a=mid" attribute matching the MID that was present in the corresponding media section from the partial offer.

If the preceding steps have been performed for each media section without resulting in a rejection, then the agent forms a partial answer consisting of a single o= line, and all of the media sections that were generated as part of the preceding steps. Note that this processing will always yield the same number of media sections in a partial answer as were present in the partial offer. Unlike normal SDP processing, however, the order of the media sections in a partial answer is not significant. This partial answer is then sent to the remote agent using the high-level protocol.

If the above processing results in a successful partial answer, then the agent’s view of the session is updated as described in Section 5.5.

5.4. Processing a Partial Answer

When a partial answer is received, the offerer matches each media section in the partial answer to its corresponding media section according to its MID. The agent MUST NOT assume that the order of the sections in the received partial answer matches the order of the sections it sent in the partial offer. However, it can expect that each section in the partial offer has a corresponding section in the receive partial answer.

For each media section, the agent then updates its local view of session state as described in Section 5.5, and follows the process described in section 7 of [RFC3264].

5.5. Updating the Shared View of Session State

Whenever a partial offer or partial answer is processed, the agent performs the following steps to ensure that a common view of session state is maintained:

1. The remote session's <sess-version> value is updated according to the value received in the o= line of the sdpfrag.

2. Any changed or removed media sections are modified in-place. Their position in the overall session description remains the same as it was before.

3. Any added media sections are appended to the existing session. The order in which they are appended is determined by lexically sorting them according to their MID values. This is not necessarily the same order in which they appear in the sdpfrag. If the recipient of a partial offer had a sent a partial offer to which it had not yet received a response when the partial offer was received, then it must take additional steps to ensure a common view of the media section ordering: the media sections for the sent partial offer and the received partial answer are treated as a single list, sorted lexically according to their respective MID values, and appended to the session in that order. When that agent receives the corresponding partial answer, the media section ordering remains the same as was established by the partial offer.

   > Note that this requires the ability to re-index media sections in the case that the remote party rejects the outstanding partial offer that we sent them (I don't mean declining the line by setting the port to zero; I mean that the higher-level protocol actually rejected the offer, like getting an unrecoverable 400- or 500-class error in SIP). OPEN ISSUE: This is kinda ugly. Is there maybe some better way to handle the issue?

5.6. Receiving a Full Offer with a Partial Offer Pending

For completeness, this document notes that an agent that receives a full offer with a sent partial offer pending is in a glare condition; this is resolved via the higher-level protocol.

6. Examples

The SDP examples given in these examples deviate from actual on-the-wire SDP notation in several ways. This is done to facilitate readability and to conform to the restrictions imposed by the RFC formatting rules. These deviations are as follows:

- Any line that is indented (compared to the initial line in the SDP block) is a continuation of the preceding line. The line break and indent are to be interpreted as a single space character.
- Empty lines in any SDP example are inserted to make functional divisions in the SDP clearer, and are not actually part of the SDP syntax.
- Excepting the above two conventions, line endings are to be interpreted as <CR><LF> pairs (that is, an ASCII 13 followed by an ASCII 10).
- Any text starting with the string "//" to the end of the line is inserted for the benefit of the reader, and is not actually part of the SDP syntax.

For the use-cases that follow, a full Offer/Answer SDP is shown followed by application of the procedures defined in this document to generate Partial Offers and Answers in carrying out the use-case.

As a pre-condition, the SDP below represents the stable state of system after a successful [RFC3264] Offer/Answer negotiation to setup a communication session with one audio (G.711) and one video (VP8) stream.

This SDP serves as base SDP for generating Offers/Partial Offers and shall be terms as Base-SDP going forward.

```
v=0
o=- 20518 0 IN IP4 203.0.113.1
s=
t=0 0
c=IN IP4 203.0.113.2
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7

m=audio 55400 RTP/SAVPF 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
```

```
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55400 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55401 typ host

m=video 55600 RTP/SAVPF  120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:98 VP8/90000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

6.1. Adding Streams

6.1.1. Full Offer/Answer Procedures

The following SDP shows an offer that adds an audio media section with Opus codec to the Base-SDP:

```
v=0
o=- 20518 1 IN IP4 198.51.100.1      // Version number is incremented
s=
t=0 0
c=IN IP4 203.0.113.1
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7

m=audio 55400 RTP/SAVPF 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55400 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55401 typ host

m=video 55600 RTP/SAVPF  120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host

m=audio 55800 RTP/SAVPF 109          // New audio media line for opus
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55800 typ host
```

```
a=candidate:1 2 UDP 2113667326 203.0.113.2 55801 typ host
```

The following shows answer for the above Offer accepting the changes:

```
v=0
o=- 20518 1 IN IP4 198.51.100.2     // Version number is incremented
s=
t=0 0
c=IN IP4 203.0.113.2
a=ice-ufrag:c300d85b
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2
a=fingerprint:sha-1
        91:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:03

m=audio 60600 RTP/SAVPF 0
a=mid:4TOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
a=sendrecv
a=candidate:0 1 UDP 2113667327 192.0.2.2 60600 typ host
a=candidate:1 2 UDP 2113667326 192.0.2.2 60401 typ host

m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host

m=audio 60604 RTP/SAVPF 109          // New audio media line for Opus
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 60604 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 60605 typ host
```

6.1.2. Partial Offer/Answer Procedures

In order to add an audio media section with Opus codec the Offerer generates the following Partial Offer:

```
o=- 20518 1 IN IP4 198.51.100.1      // Version number is incremented
m=audio 55800 RTP/SAVPF 109          // New audio media line for Opus
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
```

```
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55800 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55801 typ host
```

On receiving the above Partial Offer, the Answerer follows the validations defined in the Section 5.3 to generate a Partial Answer. Since the content of the "a=mid" attribute doesn't match any existing values and the Port Number is non zero, thus generated Partial Answer reflects accepting the new audio stream.

Below shows Partial Answer generated by the Answerer in response to the above Partial Offer.

```
o=- 20518 1 IN IP4 198.51.100.2
m=audio 60604 RTP/SAVPF 109 // Answerer accepts the new media stream.
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 60604 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 60605 typ host
```

On successful Partial Offer/Answer exchange, the Offerer appends the media section offered in the Partial Offer to its Base-SDP. Also <sess-version> is updated as per o= line received. Updated SDP at the Offerer is shown below.

```
v=0
o=- 20518 1 IN IP4 198.51.100.1  // Version number updated
s=
t=0 0
c=IN IP4 203.0.113.1
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7

m=audio 55400 RTP/SAVPF 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55400 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55401 typ host
```

```
m=video 55600 RTP/SAVPF  120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host

m=audio 55800 RTP/SAVPF 109  // Appended per Partial Offer
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55800 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55801 typ host
```

Identical steps are performed by the Answerer on the media section in the Partial Answer as shown below.

```
v=0
o=- 20518 1 IN IP4 198.51.100.2 // Version number updated
s=
t=0 0
c=IN IP4 203.0.113.2
a=ice-ufrag:c300d85b
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2
a=fingerprint:sha-1
        91:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:03

m=audio 60600 RTP/SAVPF 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
a=sendrecv
a=candidate:0 1 UDP 2113667327 192.0.2.2 60600 typ host
a=candidate:1 2 UDP 2113667326 192.0.2.2 60601 typ host

m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host

m=audio 60604 RTP/SAVPF 109 // Appended per Partial Answer
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 60604 typ host
```

```
a=candidate:1 2 UDP 2113667326 203.0.113.2 60605 typ host
```

6.2. Removing Streams

6.2.1. Full Offer/Answer Procedures

The following SDP shows an offer that removes the audio media section with PCMU Codec from the Base-SDP:

```
v=0
o=- 20518 1 IN IP4 198.51.100.1      // Version number is incremented
s=
t=0 0
c=IN IP4 203.0.113.1
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7

m=audio 0 RTP/SAVPF 0                  // Port is set to zero.
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55400 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55401 typ host

m=video 55600 RTP/SAVPF  120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:98 VP8/90000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

The following shows answer for the above Offer accepting the changes:

```
v=0
o=- 20518 1 IN IP4 198.51.100.2     // Version number is incremented
s=
t=0 0
c=IN IP4 203.0.113.2
a=ice-ufrag:c300d85b
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2
a=fingerprint:sha-1
        91:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:03
```

```
m=audio 0 RTP/SAVPF 0              // Removal of stream is accepted
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000

m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host
```

6.2.2. Partial Offer/Answer Procedures

In order to remove the audio media section from the Base-SDP the Offerer generates the following Partial Offer:

```
o=- 20518 1 IN IP4 198.51.100.1         // Version number incremented
m=audio 0 RTP/SAVPF 0                   // Port is set to zero
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW  // mid attribute is included
a=rtpmap:0 PCMU/8000
```

On receiving the above Partial Offer, the Answerer follows the validations defined in the Section 5.3 to generate a Partial Answer that accepts the removal of the corresponding media section.

Below shows the Partial Answer generated by the Answerer in response to the above Partial Offer.

```
o=- 20518 1 IN IP4 198.51.100.2
m=audio 0 RTP/SAVPF 0                // Removal of stream is accepted
a=mid:AOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
```

On successful Partial Offer/Answer exchange, the Offerer updates its Base-SDP to reflect removing of the audio media stream. Also <sess-version> is updated as per o= line received. Updated SDP at the Offerer is shown below.

```
v=0
o=- 20518 1 IN IP4 198.51.100.1 // Version number updated
```

```
s=
t=0 0
c=IN IP4 203.0.113.1
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7

m=audio 0 RTP/SAVPF 0                       // Port is updated to zero
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000

m=video 55600 RTP/SAVPF  120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:98 VP8/90000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

Identical steps are performed by the Answerer on the media section in the Partial Answer as shown below.

```
v=0
o=- 20518 1 IN IP4 198.51.100.2  // Version number updated
s=
t=0 0
c=IN IP4 203.0.113.2
a=ice-ufrag:c300d85b
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2
a=fingerprint:sha-1
        91:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:03

m=audio 0 RTP/SAVPF 0                       // Port is updated to zero
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000

m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host
```

6.3. Changing a Stream

6.3.1. Full Offer/Answer Procedures

The following SDP shows an offer that marks video stream as sendonly:

```
v=0
o=- 20518 1 IN IP4 198.51.100.1 // Version number is incremented
s=
t=0 0
c=IN IP4 203.0.113.1
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7

m=audio 55400 RTP/SAVPF 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55400 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55401 typ host

m=video 55600 RTP/SAVPF  120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=sendonly  // Video stream is marked as sendonly.
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

The following shows answer for the above Offer accepting the changes.

```
v=0
o=- 20518 1 IN IP4 198.51.100.2     // Version number is incremented
s=
t=0 0
c=IN IP4 203.0.113.2
a=ice-ufrag:c300d85b
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2
a=fingerprint:sha-1
        91:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:03

m=audio 60600 RTP/SAVPF 0
a=mid:AOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
a=sendrecv
```

```
a=candidate:0 1 UDP 2113667327 192.0.2.2 60400 typ host
a=candidate:1 2 UDP 2113667326 192.0.2.2 60401 typ host

m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=recvonly  // Answerer accepts the change and marks
            // the stream as recvonly.
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host
```

6.3.2. Partial Offer/Answer Procedures

In order to mark video stream as sendonly, an Partial Offer is generated:

```
o=- 20518 1 IN IP4 198.51.100.1  // Version number is incremented
m=video 55600 RTP/SAVPF  120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=sendonly  // Video stream is marked as sednonly.
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

Since the content of "a=mid" attribute in the Partial Offer matches, the Answerer generates an Partial Answer with media section corresponding to the video stream accepting the changes, as shown below.

```
o=- 20518 1 IN IP4 198.51.100.2
m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=recvonly   // Answerer accepts the change and marks
             // the stream as recvonly on the Answer.
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host
```

On successful Partial Offer/Answer exchange, the Offerer updates the video media section by changing the direction attribute to sendonly. Also <sess-version> is updated as per o= line received,

```
v=0
o=- 20518 1 IN IP4 198.51.100.1 // Version number updated
s=
t=0 0
c=IN IP4 203.0.113.1
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7

m=audio 55400 RTP/SAVPF 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55400 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55401 typ host

m=video 55600 RTP/SAVPF  120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=sendonly  // direction updated as per Partial O/A exchange
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

Identical steps are performed by the Answerer on the media section in the Partial Answer as shown below.

```
v=0
o=- 20518 1 IN IP4 198.51.100.2 // Version number updated
s=
t=0 0
c=IN IP4 203.0.113.2
a=ice-ufrag:c300d85b
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2
a=fingerprint:sha-1
        91:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:03

m=audio 60600 RTP/SAVPF 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
a=sendrecv
a=candidate:0 1 UDP 2113667327 192.0.2.2 60400 typ host
```

```
a=candidate:1 2 UDP 2113667326 192.0.2.2 60401 typ host

m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=recvonly  // direction updated as per Partial O/A exchange
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host
```

6.4. Both Sides Simultaneously Add Streams

Let Alice and Bob be the peers communicating.In this scenario both the parties attempt to add a new media stream at the same time.

6.4.1. Full Offer/Answer Procedures

This scenario results in the glare situation and should be resolved by the higher-level protocol.

6.4.2. Partial Offer/Answer Procedures

Alice sends a Partial Offer, shown below, to add an audio media section for Opus Codec.

```
o=- 20518 1 IN IP4 198.51.100.1      // Version number is incremented
m=audio 55800 RTP/SAVPF 109          // New audio media line for Opus
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55800 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55801 typ host
```

At the same time, Bob sends the following Partial Offer to add an video media section for H.264 Codec.

```
o=- 20518 1 IN IP4 198.51.100.2     // Version number is incremented
m=video 60604 RTP/SAVPF 99          // New video media line for H.264
a=mid:u1LS6AUZIugkXCT3S7aRFNEZOfUV18hT
a=rtpmap:99 H264/90000
a=fmtp:99 profile-level-id=4d0028;packetization-mode=1
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 60604 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60605 typ host
```

On receiving the Partial Offer from Bob, Alice verifies from the content of "a=mid" value as an indication of new media being added. It generates the Partial Answer accepting Bob’s request to add the new video stream.

```
o=- 20518 1 IN IP4 198.51.100.1
m=video 55900 RTP/SAVPF 99       // Alice accepts Bob’s Partial Offer
a=mid:u1LS6AUZIugkXCT3S7aRFNEZOfUV18hT  // MID from Partial Offer
a=rtpmap:99 H264/90000
a=fmtp:99 profile-level-id=4d0028;packetization-mode=1
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 55900 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 55901 typ host
```

Symmetrically Bob carries out similar actions on Alice’s Partial Offer and generates an Partial Answer as shown below.

```
o=- 20518 1 IN IP4 198.51.100.2
m=audio 60606 RTP/SAVPF 109      // Bob accepts Alice’s Partial Offer
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW  // MID from Partial Offer
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 60606 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 60607 typ host
```

On successful Partial Offer/Answer exchange, Alice appends to the Base-SDP, the two media sections that correspond to audio and video streams negotiated as part of aforementioned Partial Offer/Answer exchanges. Also <sess-version> is incremented to reflect the shared state. The media sections are appended in the lexically increasing order.

```
v=0
o=- 20518 2 IN IP4 198.51.100.1 // Version number updated
s=
t=0 0
c=IN IP4 203.0.113.1
a=ice-ufrag:F7gI
a=ice-pwd:x9cml/YzichV2+XlhiMu8g
a=fingerprint:sha-1
        42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7
```

```
m=audio 55400 RTP/SAVPF 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55400 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55401 typ host

m=video 55600 RTP/SAVPF  120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host

m=audio 55800 RTP/SAVPF 109  // New audio media line for Opus
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 55800 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55801 typ host

m=video 55900 RTP/SAVPF 99  // Alice accepts Bob's Partial Offer
a=mid:u1LS6AUZIugkXCT3S7aRFNEZOfUV18hT
a=rtpmap:99 H264/90000
a=fmtp:99 profile-level-id=4d0028;packetization-mode=1
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 55900 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 55901 typ host
```

Similarly, below SDP shows Bob's Base-SDP updated.

```
v=0
o=- 20518 2 IN IP4 198.51.100.2  // Version number updated
s=
t=0 0
c=IN IP4 203.0.113.2
a=ice-ufrag:c300d85b
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2
a=fingerprint:sha-1
        91:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:03

m=audio 60600 RTP/SAVPF 0
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:0 PCMU/8000
a=sendrecv
a=candidate:0 1 UDP 2113667327 192.0.2.2 60600 typ host
```

```
a=candidate:1 2 UDP 2113667326 192.0.2.2 60601 typ host

m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host

m=audio 60606 RTP/SAVPF 109      // Bob accepts Alice's Partial Offer
a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
a=rtpmap:109 opus/48000/2
a=sendrecv
a=candidate:0 1 UDP 2113667327 203.0.113.2 60606 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 60607 typ host

m=video 60604 RTP/SAVPF 99          // New video media line for H.264
a=mid:u1LS6AUZIugkXCT3S7aRFNEZOfUV18hT
a=rtpmap:99 H264/90000
a=fmtp:99 profile-level-id=4d0028;packetization-mode=1
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 60604 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60605 typ host
```

6.5. Removing a Stream with Pseudo-Glare

In this example, Alice attempts to change the direction of the video stream to recvonly and Bob attempts to de-activate the same video stream simultaneously.

6.5.1. Full Offer/Answer Procedures

This scenario results in the glare situation and should be resolved by the higher-level protocol.

6.5.2. Partial Offer/Answer Procedures

The term pseudo-glare signifies those scenarios wherein both parties attempt to operate on a stream at the same time, but the final session state can be unambiguously resolved by both sides without any further signaling.

Such an scenario arises when one side tries to change a media section and simultaneously the other party attempts to remove that media section.

Below represents the Partial Offer from Alice to change the direction attribute of the video section to recvonly.

```
o=- 20518 1 IN IP4 198.51.100.1   // Version number is incremented
m=video 55600 RTP/SAVPF  120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=recvonly      // direction changed to recvonly
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

At the same time, Bob sends the following Partial Offer to remove the same media section:

```
o=- 20518 1 IN IP4 198.51.100.2     // Version number is incremented
m=video 0 RTP/SAVPF 120             // Port number is set to 0.
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
```

On validating the Partial Offer from Alice, Bob concludes the media section matches the one in the Partial Offer sent by him. By following the procedures in Section 5.3, Bob rejects Alice Partial Offer since the video media section will be disabled momentarily.

Bob sends the Partial Answer by setting port number to zero as response to Alice’s Partial Offer.

```
o=- 20518 1 IN IP4 198.51.100.1     // Version number is incremented
m=video 0 RTP/SAVPF  120            // Alice’s Partial Offer rejected
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
```

The processing by Alice is insignificant, because it will eventually be overtaken by Bob’s rejection of her Partial Offer and thus the Partial Offer/Answer exchange concludes by removing the video section.

Finally the Base-SDPs are updated by both the parties ending up in the shared state.

```
   // Alice's Base-SDP updated
   v=0
   o=- 20518 1 IN IP4 198.51.100.1 // Version number updated
   s=
   t=0 0
   c=IN IP4 203.0.113.1
   a=ice-ufrag:F7gI
   a=ice-pwd:x9cml/YzichV2+XlhiMu8g
   a=fingerprint:sha-1
           42:89:c5:c6:55:9d:6e:c8:e8:83:55:2a:39:f9:b6:eb:e9:a3:a9:e7

   m=audio 55400 RTP/SAVPF 0
   a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
   a=rtpmap:0 PCMU/8000
   a=sendrecv
   a=candidate:0 1 UDP 2113667327 203.0.113.2 55400 typ host
   a=candidate:1 2 UDP 2113667326 203.0.113.2 55401 typ host

   m=video 0 RTP/SAVPF  120          // Video stream removed
   a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
   a=rtpmap:120 VP8/90000
   a=sendrecv
   a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
   a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host



   // Bob's Base-SDP
   v=0
   o=- 20518 1 IN IP4 198.51.100.2  // Version number updated.
   s=
   t=0 0
   c=IN IP4 203.0.113.2
   a=ice-ufrag:c300d85b
   a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2
   a=fingerprint:sha-1
           91:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:03

   m=audio 60600 RTP/SAVPF 0
   a=mid:ATOnU45h09BqsacSCyQwuFttyBkSFQGW
   a=rtpmap:0 PCMU/8000
   a=sendrecv
   a=candidate:0 1 UDP 2113667327 192.0.2.2 60600 typ host
   a=candidate:1 2 UDP 2113667326 192.0.2.2 60601 typ host

   m=video 0 RTP/SAVPF 120  // Port is zero per Bob's Partial Offer
   a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
   a=rtpmap:120 VP8/90000
```

```
a=sendrecv
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host
```

6.6. Changing a Stream with Glare

In this example both Alice and Bob attempt to update the same media section that conflicts each others actions.

6.6.1. Full Offer/Answer Procedures

This scenario results in the glare situation and should be resolved by the higher-level protocol.

6.6.2. Partial Offer/Answer Procedures

To explain this scenario, say Alice attempts to update the video section’s direction to be sendonly and Bob also attempts to perform the same action.

This results in a conflict situation since both the parties can’t have the same media section with sendonly direction, since it violates rules defined in [RFC3264]

Partial Offer’s generated by Alice and Bob for the above scenario is shown below.

```
// Alice’s Partial Offer
o=- 20518 1 IN IP4 198.51.100.1      // Version number is incremented
m=video 55600 RTP/SAVPF  120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=sendonly                          // direction changed to sendonly
a=candidate:0 1 UDP 2113667327 203.0.113.2 55600 typ host
a=candidate:1 2 UDP 2113667326 203.0.113.2 55601 typ host
```

```
// Bob’s Partial Offer
o=- 20518 1 IN IP4 198.51.100.1     // Version number is incremented
m=video 60602 RTP/SAVPF 120
a=mid:0Ny4mOBV2MWTH1JYRRNORarcTbG11QxV
a=rtpmap:120 VP8/90000
a=sendonly                        // direction changed to sendonly
```

```
a=candidate:2 1 UDP 2113667327 192.0.2.2 60602 typ host
a=candidate:3 2 UDP 2113667326 192.0.2.2 60603 typ host
```

This results in a glare situation under Partial Offer/Answer exchange due to the conflicting nature of the actions. To resolve this situation assistance from the higher level application protocol is required.

7. Security Considerations

TBD

8. IANA Considerations

TBD -- I don’t think we actually need any for this mechanism.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.

9.2. Informative References

[I-D.ivov-dispatch-sdpfrag] Ivov, E. and A. Roach, "Internet Media Type application/sdpfrag", draft-ivov-dispatch-sdpfrag-03 (work in progress), October 2013.

[RFC3311] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.

[RFC6086] Holmberg, C., Burger, E., and H. Kaplan, "Session Initiation Protocol (SIP) INFO Method and Package Framework", RFC 6086, January 2011.

Authors' Addresses

Adam Roach
Mozilla
Dallas, TX
US

Phone: +1 650 903 0800 x863
Email: adam@nostrum.com

Suhas Nandakumar
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: snandaku@cisco.com