MMUSIC   K. Drage, Ed.
Internet-Draft   M. Makaraju
Intended status: Standards Track   J. Stoetzer-Bradler
Expires: April 30, 2015   Alcatel-Lucent
                                                                   R. Ejzak
                                                                  J. Marcon
                                                               Unaffiliated
                                                           October 27, 2014

                SDP-based "SCTP over DTLS" data channel negotiation
                    draft-ejzak-mmusic-data-channel-sdpneg-02

Abstract

   The Real-Time Communication in WEB-browsers (RTCWeb) working group is
   charged to provide protocols to support direct interactive rich
   communications using audio, video, and data between two peers' web-
   browsers. For the support of data communication, the RTCWeb working
   group has in particular defined the concept of bi-directional data
   channels over SCTP, where each data channel might be used to
   transport other protocols, called sub-protocols. Data channel setup
   can be done using either the internal in-band band (also referred to
   as 'internal' for the rest of the document) WebRTC Data Channel
   Establishment Protocol or some external out-of-band simply referred
   to as 'external negotiation' in the rest of the document . This
   document specifies how the SDP offer/answer exchange can be used to
   achieve such an external negotiation. Even though data channels are
   designed for RTCWeb use initially they may be used by other protocols
   like, but not limited to, the CLUE protocol. This document is
   intended to be used wherever data channels are used.

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 30, 2015.

Copyright Notice

Table of Contents

1. Introduction

   The RTCWeb working group has defined the concept of bi-directional
   data channels running on top of SCTP/DTLS. RTCWeb leaves it open for
   other applications to use data channels and its in-band or out-of-
   band protocol for creating them. Each data channel consists of
   paired SCTP streams sharing the same SCTP Stream Identifier. Data
   channels are created by endpoint applications through the WebRTC API,
   or other users of data channel like CLUE, and can be used to
   transport proprietary or well-defined protocols, which in the latter
   case can be signaled by the data channel "sub-protocol" parameter,
   conceptually similar to the WebSocket "sub-protocol". However, apart
   from the "sub-protocol" value transmitted to the peer, RTCWeb leaves
   it open how endpoint applications can agree on how to instantiate a
   given sub-protocol on a data channel, and whether it is signaled in-
   band or out-of-band (or both). In particular, the SDP offer
   generated by the application includes no channel-specific
   information.

   This document defines SDP-based out-of-band negotiation procedures to
   establish data channels for transport of well-defined sub-protocols.

2. Conventions

   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. Terminology

   This document uses the following terms:

      Data channel: A bidirectional channel consisting of paired SCTP
      outbound and inbound streams.

      Data channel stack: An entity which, upon application request,
      runs data channel protocol to keep track of states, sending and
      receive data. If the application is browser based Javascript
      application then this stack resides in the browser. If the
      application is a native application then this stack resides in
      application and accessible to it via some sort of APIs.

Data channel properties: fixed properties assigned to a data
channel at the time of its creation. Some of these properties
determine the way the data channel stack transmits data on this
channel (e.g., stream identifier, reliability, order of
delivery...)

DCEP - Data Channel Establishment Protocol defined in
[I-D.ietf-rtcweb-data-protocol].

External negotiation: Data channel negotiation based on SDP offer/
answer outlined in this specification.

Internal negotiation: Data channel negotiation based on Data
Channel Establishment Protocol defined in
[I-D.ietf-rtcweb-data-protocol].

In-band: transmission through the peer-to-peer SCTP association.

In-band negotiation: data channel negotiation based Data Channel
Establishment Protocol defined in [I-D.ietf-rtcweb-data-protocol].

Out-of-band: transmission through the application signaling path.

Peer: From the perspective of one of the agents in a session, its
peer is the other agent. Specifically, from the perspective of
the SDP offerer, the peer is the SDP answerer. From the
perspective of the SDP answerer, the peer is the SDP offerer.

Stream identifier: the identifier of the outbound and inbound SCTP
streams composing a data channel.

4. Data Channels

This section summarizes how data channels work in general. Note that
the references to 'browser' here is intentional as in this specific
example the data channel user is a webrtc enabled browser.

A WebRTC application creates a data channel via the Data Channel API,
by providing a number of setup parameters (sub-protocol, label,
reliability, order of delivery, priority). The application also
specifies if it wants to make use of the in-band negotiation using
the DCEP [I-D.ietf-rtcweb-data-protocol], or if the application
intends to perform an "external negotiation" using some other in-band
or out-of-band mechanism.

In any case, the SDP offer generated by the browser is per
[I-D.ietf-mmusic-sctp-sdp]. In brief, it contains one m-line for the

SCTP association on top of which data channels will run, and one
attribute per protocol assigned to the SCTP ports:

OPEN ISSUE: The syntax in [I-D.ietf-mmusic-sctp-sdp] may change as
that document progresses. In particular we expect "webrtc-
datachannel" to become a more general term.

```
m=application 54111 DTLS/SCTP webrtc-datachannel
c=IN IP4 79.97.215.79
a=fmtp:webrtc-datachannel max-message-size=100000
a=sctp-port 5000
a=setup:actpass
a=connection:new
a=fingerprint:SHA-1 \
    4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
```

Note: A WebRTC browser will only use m-line format "webrtc-
datachannel", and will not use other formats in the m-line for other
protocols such as t38. [I-D.ietf-mmusic-sctp-sdp] supports only one
SCTP association to be established on top of a DTLS session.

Note: This SDP syntax does not contain any channel-specific
information.

4.1. Stream identifier numbering

Independently from the requested type of negotiation, the application
creating a data channel can either pass to the browser the stream
identifier to assign to the data channel or else let the browser pick
one identifier from the ones unused.

To avoid glare situations, each endpoint can moreover own an
exclusive set of stream identifiers, in which case an endpoint can
only create a data channel with a stream identifier it owns.

Which set of stream identifiers is owned by which endpoint is
determined by convention or other means.

   For data channels negotiated in-band, one endpoint owns by
   convention the even stream identifiers, whereas the other owns the
   odd stream identifiers, as defined in
   [I-D.ietf-rtcweb-data-protocol].

   For data channels externally negotiated, no convention is defined
   by default.

4.2. Generic external negotiation

4.2.1. Overview

In-band negotiation only provides for negotiation of data channel transport parameters and does not provide for negotiation of sub-protocol specific parameters. External negotiation can be defined to allow negotiation of parameters beyond those handled by in-band negotiation, e.g., parameters specific to the sub-protocol instantiated on a particular data channel. See Section 5.1.2 for an example of such a parameter.

The following procedures are common to all methods of external negotiation, whether in-band (communicated using proprietary means on an already established data channel) or out-of-band (using SDP or some other protocol associated with the signaling channel).

4.2.2. Opening a data channel

In the case of external negotiation, the endpoint application has the option to fully control the stream identifier assignments. However these assignments have to coexist with the assignments controlled by the data channel stack for the in-band negotiated data channels (if any). It is the responsibility of the application to ensure consistent assignment of stream identifiers.

When the application requests the creation of a new data channel to be set up via external negotiation, the data channel stack creates the data channel locally without sending any DATA CHANNEL OPEN message in-band, and sets the data channel state to Connecting if the SCTP association is not yet established, or sets the data channel state to Open if the SCTP association is already established. The side which starts external negotiation creates data channel using underlying data channel stack API and the data channel is put into open state immediately (assuming ICE, SCTP procedures were already done). However, the application can't send data on this data channel until external negotiation is complete with the peer. This is because peer needs to be aware and accept the data channel via external negotiation. The peer after accepting the data channel offer can start sending data immediately. This implies that offerer may get data channel message before external negotiation is complete and the application should be ready to handle it.

If the peer rejects the data channel part of the offer then it doesn't have to do anything as the data channel was not created using the stack. The offerer on the other hand needs to close the data channel that was opened by invoking relevant data channel stack API procedures.

It is also worth noting that a data channel stack implementation may not provide any API to create and close data channels; instead the data channels are used on the fly as needed just by communicating via external means or by even having some local configuration/assumptions on both the peers.

The application then externally negotiates the data channel properties and sub-protocol properties with the peer's application.

[ASSUMPTION] The peer must then symmetrically create a data channel with these negotiated data channel properties. This is the only way for the peer's data channel stack to know which properties to apply when transmitting data on this channel. The data channel stack must allow data channel creation with any non-conflicting stream identifier so that both peers can create the data channel with the same stream identifier.

In case the external negotiation is correlated with an SDP offer/ answer exchange that establishes the SCTP association, the SCTP initialization completion triggers a callback from the data channel stack to an application on both the ends to change the data channel state from Connecting to Open. The details of this interface is specific to the data channel user application. Browser based applications (could include hybrid apps) will use [WebRtcAPI], while native applications use a compatible API, which is yet to be specified. See Section 5.2.2 for details on when the data channel stack can assume the data channel is open, and on when the application can assume the data channel is open.

4.2.3. Closing a data channel

When the application requests the closing of an externally negotiated data channel, the data channel stack always performs an in-band SSN reset for this channel.

Depending upon the method used for external negotiation and the sub- protocol associated with the data channel, the closing might in addition be signaled to the peer via external negotiation.

5. SDP-based external negotiation

This section defines a method of external negotiation by which two clients can negotiate data channel-specific and sub-protocol-specific parameters, using the out-of-band SDP offer/answer exchange. This SDP extension can only be used with SDP offer/answer model.

5.1. SDP syntax

Two new SDP attributes are defined to support external negotiation of
data channels. The first attribute provides for negotiation of
channel-specific parameters. The second attribute provides for
negotiation of sub-protocol-specific parameters.

5.1.1. SDP attribute for data channel parameter negotiation

Associated with the SDP "m" line that defines the SCTP association
for data channels (defined in Section 4), each SDP offer and answer
includes an attribute line that defines the data channel parameters
for each data channel to be negotiated. Each attribute line
specifies the following parameters for a data channel: Stream
Identifier, sub-protocol, label, reliability, order of delivery, and
priority. Conveying a reliable data channel is achieved by including
neither 'max-retr' nor 'max-time'. Conveying an unreliable data
channel is achieved by including only one of 'max-retr' or 'max-
time'. By definition max-retr and max-time are mutually exclusive,
so only one of them can be present in a=dcmap. If an SDP offer
contains both of these parameters then such an SDP offer will be
rejected. If an SDP answer contains both of these parameters then
the offerer may treat it as an error and may assume the associated
SDP offer/answer failed and may take appropriate recovery actions.
These recovery options are outside the scope of this specification.
Following is an example of the attribute line for sub-protocol "BFCP"
and stream id "2":

a=dcmap:2 subprotocol="BFCP";label="channel 2"

The SDP answer shall echo the same subprotocol, max-retr, max-time,
ordered parameters, if those were present in the offer, and may
include a label parameter. They may appear in any order, which could
be different from the SDP offer, in the SDP answer.

The same information MUST be replicated without changes in any
subsequent offer or answer, as long as the data channel is still
opened at the time of offer or answer generation.

   Note: This attribute is derived from attribute "webrtc-
   DataChannel", which was defined in old version 03 of the following
   draft, but which was removed along with any support for SDP
   external negotiation in subsequent versions:
   [I-D.ietf-mmusic-sctp-sdp].

   Note: This document does not provide a complete specification of
   how to negotiate the use of a data channel to transport BFCP.
   Procedures specific to each sub-protocol such as BFCP will be

documented elsewhere. The use of BFCP is only an example of how
the generic procedures described herein might apply to a specific
sub-protocol.

The intention of exchanging these attributes is to create data
channels on both the peers with the same set of attributes without
actually using [I-D.ietf-rtcweb-data-protocol]. It is assumed that
the data channel properties (reliable/unreliable, ordered/unordered)
are suitable per the sub-protocol transport requirements. Data
channel types defined in [I-D.ietf-rtcweb-data-protocol] are mapped
to SDP in the following manner:

```
DATA_CHANNEL_RELIABLE
      a=dcmap:2 subprotocol="BFCP";label="channel 2"

DATA_CHANNEL_RELIABLE_UNORDERED
      a=dcmap:2 subprotocol="BFCP";label="channel 2";\
            ordered=0

DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT
      a=dcmap:2 subprotocol="BFCP";label="channel 2";\
            max-retr=3

DATA_CHANNEL_PARTIAL_RELIABLE_REXMIT_UNORDERED
      a=dcmap:2 subprotocol="BFCP";label="channel 2";\
            max-retr=3;ordered=0;

DATA_CHANNEL_PARTIAL_RELIABLE_TIMED
      a=dcmap:2 subprotocol="BFCP";label="channel 2";\
            max-time=10000;

DATA_CHANNEL_PARTIAL_RELIABLE_TIMED_UNORDERED
      a=dcmap:2 subprotocol="BFCP";label="channel 2";\
            max-time=10000; ordered=0
```

5.1.1.1. dcmap attribute

The 'stream' parameter indicates the actual stream identifier within
the association used to form the channel. Stream is a mandatory
parameter and is noted directly after the "a=dcmap:" attribute's
colon.

Formal Syntax:
TBD: Should this be moved to SDP grammar section?

Name: dcmap

Value: dcmap-value

Usage Level: media

Charset Dependent: no

Syntax:

```
dcmap-value      = dcmap-stream-id
                   [ SP dcmap-opt *(";" dcmap-opt) ]
dcmap-opt        = ordering-opt / subprotocol-opt / label-opt
                   / maxretr-opt / maxtime-opt
                   ; Either only maxretr-opt or maxtime-opt
                   ; is present.
                   ; Both MUST not be present.

dcmap-stream-id = 1*DIGIT
ordering-opt    = "ordered=" ordering-value
ordering-value  = "0"/"1"
subprotocol-opt = "subprotocol=" quoted-string
label-opt       = "label=" quoted-string
maxretr-opt     = "max-retr=" maxretr-value
maxretr-value   = <from-Reliability-Parameter of
                    I-D.ietf-rtcweb-data-protocol>
                   ; number of retransmissions
maxtime-opt     = "max-time=" maxtime-value
maxtime-value   = <from-Reliability-Parameter of
                   I-D.ietf-rtcweb-data-protocol>
                   ; milliseconds

quoted-string   = DQUOTE *(quoted-char / escaped-char) DQUOTE
quoted-char     = SP / quoted-visible
quoted-visible  = %21 / %23-24 / %26-7E ; VCHAR without " or %
escaped         = "%" HEXDIG HEXDIG
DQUOTE          = <from-RFC5234>
integer         = <from-RFC5234>
```

Examples:

```
a=dcmap:0
a=dcmap:1 subprotocol="BFCP";max-time=60000
a=dcmap:2 subprotocol="MSRP";ordered;label="MSRP"
a=dcmap:3 label="Label 1";unordered;max-retr=5
a=dcmap:4 label="foo%09bar";ordered;max-time=15000;max-retr=3
```

5.1.1.2. label parameter

The optional 'label' parameter indicates the name of the channel. It
represents a label that can be used to distinguish, in the context of
the WebRTC API, an RTCDataChannel object from other RTCDataChannel

objects. Label is a mandatory parameter. This parameter maps to the
'label' parameter defined in [I-D.ietf-rtcweb-data-protocol]

5.1.1.3. subprotocol parameter

The 'subprotocol' parameter indicates which protocol the client
expects to exchange via the channel. Subprotocol is a mandatory
parameter.

[ACTION ITEM] The IANA registry to be used for the subprotocol
parameter is still to be determined. It also needs to be determined
what the relationship is to existing registries and how to reference
already-existing protocols.

5.1.1.4. max-retr parameter

This parameter indicates that the data channel is unreliable. The
'max-retr' parameter indicates the max times a user message will be
retransmitted. The max-retr parameter is optional with default value
unbounded. This parameter maps to the 'Number of RTX' parameter
defined in [I-D.ietf-rtcweb-data-protocol]

5.1.1.5. max-time parameter

This parameter indicates that the data channel is unreliable. A user
messages will no longer be transmitted or retransmitted after a
specified life-time given in milliseconds in the 'max-time'
parameter. The max-time parameter is optional with default value
unbounded. This parameter maps to the 'Lifetime in ms' parameter
defined in [I-D.ietf-rtcweb-data-protocol]

5.1.1.6. ordered parameter

The ordered' parameter indicates that DATA chunks in the channel MUST
be dispatched to the upper layer by the receiver while preserving the
order. The ordered parameter is optional and takes two values: "0"
for ordered and "1" for ordered delivery with "1" as the default
value. Any other value is ignored and default ordered is assumed.
If the ordered parameter is absent, the receiver is required to
deliver DATA chunks to the upper layer in proper order. This
parameter maps to the ordered or unorderd data channel types as
defined in [I-D.ietf-rtcweb-data-protocol]

5.1.2. Sub-protocol specific attributes

In the SDP, each data channel declaration MAY also be followed by
other SDP attributes specific to the sub-protocol in use. Each of
these attributes is represented by one new attribute line, and it

includes the contents of a media-level SDP attribute already defined
for use with this (sub)protocol in another IETF specification. Sub-
protocol-specific attributes might also be defined for exclusive use
with data channel transport, but should use the same syntax described
here for other sub-protocol-specific attributes.

Each sub-protocol specific SDP attribute that would normally be used
to negotiate the subprotocol using SDP is replaced with an attribute
of the form "a=dcsa: stream-id original-attribute", where dcsa stands
for "data channel sub-protocol attribute", stream-id is the sctp
stream identifier assigned to this sub-protocol instance, and
original-attribute represents the contents of the sub-protocol
related attribute to be included.

Formal Syntax:

Name: dcsa

Value: dcsa-value

Usage Level: media

Charset Dependent: no

Syntax:

dcsa-value   = stream-id SP attribute
attribute   = <from-RFC4566>

Examples:

a=dcsa:2 accept-types:text/plain

Thus in the example above, the original attribute line "a=accept-
types:text/plain" is represented by the attribute line "a=dcsa:2
accept-types:text/plain", which specifies that this instance of MSRP
being transported on the sctp association using the data channel with
stream id 2 accepts plain text files. The above example creates a
reliable, ordered data channel.

As opposed to the data channel setup parameters, these parameters are
subject to offer/answer negotiation following the procedures defined
in the sub-protocol specific documents.

The same syntax applies to any other SDP attribute required for
negotiation of this instance of the sub-protocol.

   Note: This document does not provide a complete specification of how
   to negotiate the use of a data channel to transport MSRP. Procedures
   specific to each sub-protocol such as MSRP will be documented
   elsewhere. The use of MSRP is only an example of how the generic
   procedures described herein might apply to a specific sub-protocol.

5.2. Procedures

5.2.1. Managing stream identifiers

   For the SDP-based external negotiation described in this document,
   the initial offerer based "SCTP over DTLS" owns by convention the
   even stream identifiers whereas the initial answerer owns the odd
   stream identifiers. This ownership is invariant for the whole
   lifetime of the signaling session, e.g. it does not change if the
   initial answerer sends a new offer to the initial offerer.

   This specification allows simultaneous use of external and internal
   negotiation. However, a single stream is managed using one method at
   a time. Stream ids that are not currently used in SDP can be used
   for internal negotiation. Stream id allocation per SDP based
   external negotiation may not align with DTLS role based allocation.
   This could cause glare conditions when one side trying to do external
   negotiation on a stream id while the other end trying to open data
   channel on the same stream id using internal negotiation. To avoid
   these glare conditions this specification recommends that the data
   channel stack user always selects stream ids per SDP offer/answer
   rule even when internal negotiation is used. To avoid glare
   conditions, it is possible to come up with a different stream id
   allocation scheme, but such schemes are outside the scope of this
   specification.

5.2.2. Opening a data channel

   The procedure for opening a data channel using external negotiation
   starts with the agent preparing to send an SDP offer. If a peer
   receives an SDP offer before getting to send a new SDP offer with
   data channels that are to be externally negotiated, or loses an SDP
   offer glare resolution procedure in this case, it must wait until the
   ongoing SDP offer/answer completes before resuming the external
   negotiation procedure.

   The agent that intends to send an SDP offer to create data channels
   through SDP-based external negotiation performs the following:

   o Creates data channels using stream identifiers from the owned set
      (see Section 5.2.1).

o As described in Section 4.2.2, if the SCTP association is not yet
   established, then the newly created data channels are in the
   Connecting state, else if the SCTP association is already
   established, then the newly created data channels are in the Open
   state.

o Generates a new SDP offer. In the case of the browser based
   applications the browser generates the offer via the createOffer()
   API call [I-D.ietf-rtcweb-jsep].

o Determines the list of stream identifiers assigned to data
   channels opened through external negotiation.

o Completes the SDP offer with the dcmap and dcsa attributes needed,
   if any, for each externally-negotiated data channel, as described
   in Section 5.1.

o Sends the SDP offer.

The peer receiving such an SDP offer performs the following:

o Applies the SDP offer. Note that the browser ignores data channel
   specific attributes in the SDP.

o Analyzes the channel parameters and sub-protocol attributes to
   determine whether to accept each offered data channel.

o For accepted data channels, creates peer instances for the data
   channels with the browser using the channel parameters described
   in the SDP offer. Note that the browser is asked to create data
   channels with stream identifiers not "owned" by the agent.

o As described in Section 4.2.2, if the SCTP association is not yet
   established, then the newly created data channels are in the
   Connecting state, else if the SCTP association is already
   established, then the newly created data channels are in the Open
   state.

o Generates an SDP answer.

o Completes the SDP answer with the dcmap and optional dcsa
   attributes needed for each externally-negotiated data channel, as
   described in Section 5.1.

o Sends the SDP answer.

The agent receiving such an SDP answer performs the following:

o Closes any created data channels (whether in Connecting or Open
   state) for which the expected dcmap and dcsa attributes are not
   present in the SDP answer.

o Applies the SDP answer.

Any data channels in Connecting state are transitioned to the Open
state when the SCTP association is established.

Each agent application MUST wait to send data until it has
confirmation that the data channel at the peer is in the Open state.
For webrtc, this is when both data channel stacks have channel
parameters instantiated. This occurs:

o At both peers when a data channel is created without an
   established SCTP association, as soon as the data channel stacks
   report that the data channel transitions to the Open state from
   the Connecting state.

o At the agent receiving an SDP offer for which there is an
   established SCTP association, as soon as it creates an externally
   negotiated data channel in the Open state based on information
   signaled in the SDP offer.

o At the agent sending an SDP offer to create a new externally
   negotiated data channel for which there is an established SCTP
   association, when it receives the SDP answer confirming acceptance
   of the data channel or when it begins to receive data on the data
   channel from the peer, whichever occurs first.

5.2.3. Closing a data channel

When the application requests the closing of a data channel that was
externally negotiated, the data channel stack always performs an in-
band SSN reset for this channel.

It is specific to the sub-protocol whether this closing must in
addition be signaled to the peer via a new SDP offer/answer exchange.

A data channel can be closed by sending a new SDP offer which
excludes the dcmap and dcsa attributes lines for the data channel.
The port value for the m line should not be changed (e.g., to zero)
when closing a data channel (unless all data channels are being
closed and the SCTP association is no longer needed), since this
would close the SCTP association and impact all of the data channels.
If answerer accepts the SDP offer then it MUST also exclude the
corresponding attribute lines in the answer. In addition to that,
SDP answerer may exclude other data channels which were closed but

not yet communicated to the peer. So, offerer MUST inspect the
answer to see if it has to close other data channels which are now
not included in the answer

If a new SDP offer/answer is used to close data channels then the
data channel(s) should only be closed by the answerer/offerer after
successful SDP answer is sent/received.

   This delayed close is to handle cases where a successful SDP
   answer is not received, in which case the state of session should
   be kept per the last successful SDP offer/answer.

If a client receives a data channel close indication (due to inband
SSN reset or some other reason) without associated SDP offer then an
SDP offer which excludes this closed data channel SHOULD be
generated.

The application must also close any data channel that was externally
negotiated, for which the stream identifiers are not listed in an
incoming SDP offer.

A closed data channel using local close (SCTP reset), without an
additional SDP offer/answer to close it, may be reused for a new data
channel. This can only be done via new SDP offer/answer, describing
the new sub-protocol and its attributes, only after the corresponding
data channel close acknowledgement is received from the peer (i.e.
SCTP reset of both incoming and outgoing streams is completed). This
restriction is to avoid the race conditions between arrival of "SDP
offer which reuses stream" with "SCTP reset which closes outgoing
stream" at the peer

5.2.4. Various SDP offer/answer scenarios and considerations

   SDP offer has no a=dcmap attributes

   * Initial SDP offer: No data channel negotiated yet.

   * Subsequent SDP offer: All the externally negotiated data
      channels must be closed now. The DTLS/SCTP association remains
      open for external or internal negotiation of data channels.

   SDP answer has no a=dcmap attributes

   * Initial SDP answer: Either the peer does not support dcmap
      attributes or it rejected all the data channels. In either
      case offerer closes all the externally negotiated data channels
      that were open at the time of initial offer. The DTLS/SCTP
      association will still be setup.

        *   Sub-sequent SDP answer: All the externally negotiated data
            channels must be closed now. The DTLS/SCTP association remains
            open for future external or internal negotiation of data
            channels.

     SDP offer has no a=dcsa attributes for a data channel.

        *   This is allowed and indicates there are no sub-protocol
            parameters to convey.

     SDP answer has no a=dcsa attributes for a data channel.

        *   This is allowed and indicates there are no sub-protocol
            parameters to convey in the SDP answer. The number of dcsa
            attributes in the SDP answer does not have to match the number
            of dcsa attributes in the SDP offer.

6. Examples

     SDP offer:
       m=application 10001 DTLS/SCTP webrtc-datachannel
       c=IN IP4 10.10.10.1
       a=fmtp:webrtc-datachannel max-message-size=100000
       a=sctp-port 5000
       a=setup:actpass
       a=connection:new
       a=fingerprint:SHA-1 \
            4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
       a=dcmap:0 subprotocol="BFCP";label="BGCP"

     SDP answer:
       m=application 10002 DTLS/SCTP webrtc-datachannel
       c=IN IP4 10.10.10.2
       a=fmtp:webrtc-datachannel max-message-size=100000
       a=sctp-port 5002
       a=setup:passive
       a=connection:new
       a=fingerprint:SHA-1 \
            5B:AD:67:B1:3E:82:AC:3B:90:02:B1:DF:12:5D:CA:6B:3F:E5:54:FA

                            Figure 1: Example 1

     In the above example the SDP answerer rejected the data channel with
     stream id 0 either for explicit reasons or because it does not
     understand the a=dcmap attribute. As a result the offerer will close
     the data channel created with the external negotiation option. The
     SCTP association will still be setup over DTLS. At this point

offerer or answerer may use internal negotiation to open data
channels.

```
SDP offer:
  m=application 10001 DTLS/SCTP webrtc-datachannel
  c=IN IP4 10.10.10.1
  a=fmtp:webrtc-datachannel max-message-size=100000
  a=sctp-port 5000
  a=setup:actpass
  a=connection:new
  a=fingerprint:SHA-1 \
      4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
  a=dcmap:0 subprotocol="BFCP";label="BGCP"
  a=dcmap:2 subprotocol="MSRP";label="MSRP"
  a=dcsa:2 accept-types:message/cpim text/plain text/
  a=dcsa:2 path:msrp://alice.example.com:10001/2s93i93idj;dc

SDP answer:
  m=application 10002 DTLS/SCTP webrtc-datachannel
  c=IN IP4 10.10.10.2
  a=fmtp:webrtc-datachannel max-message-size=100000
  a=sctp-port 5002
  a=setup:passive
  a=connection:new
  a=fingerprint:SHA-1 \
      5B:AD:67:B1:3E:82:AC:3B:90:02:B1:DF:12:5D:CA:6B:3F:E5:54:FA
  a=dcmap:2 subprotocol="MSRP";label="MSRP"
  a=dcsa:2 accept-types:message/cpim text/plain
  a=dcsa:2 path:msrp://bob.example.com:10002/si438dsaodes;dc
```

Figure 2: Example 2

In the above example SDP offer contains data channels for BFCP and
MSRP sub-protocols. SDP answer rejected BFCP and accepted MSRP. So,
the offerer should close the data channel for BFCP and both offerer
and answerer may start using MSRP data channel (after SCTP/DTLS
association is setup). The data channel with stream id 0 is free and
can be used for future internal or external negotiation.

Continuing on the earlier example in Figure 1.

```
   Subsequent SDP offer:
     m=application 10001 DTLS/SCTP webrtc-datachannel
     c=IN IP4 10.10.10.1
     a=fmtp:webrtc-datachannel max-message-size=100000
     a=sctp-port 5000
     a=setup:actpass
     a=connection:existing
     a=fingerprint:SHA-1 \
         4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
     a=dcmap:4 subprotocol="MSRP";label="MSRP"
     a=dcsa:4 accept-types:message/cpim text/plain
     a=dcsa:4 path:msrp://alice.example.com:10001/2s93i93idj;dc

   Subsequent SDP answer:
     m=application 10002 DTLS/SCTP webrtc-datachannel
     c=IN IP4 10.10.10.2
     a=fmtp:webrtc-datachannel max-message-size=100000
     a=sctp-port 5002
     a=setup:passive
     a=connection:existing
     a=fingerprint:SHA-1 \
         5B:AD:67:B1:3E:82:AC:3B:90:02:B1:DF:12:5D:CA:6B:3F:E5:54:FA
     a=dcmap:4 subprotocol="MSRP";label="MSRP"
     a=dcsa:4 accept-types:message/cpim text/plain
     a=dcsa:4 path:msrp://bob.example.com:10002/si438dsaodes;dc
```

                        Figure 3: Example 3

   The above example is a continuation of the example in Figure 1. The
   SDP offer now removes the MSRP data channel with stream id 2, but
   opens a new MSRP data channel with stream id 4. The answerer
   accepted the entire offer. As a result the offerer closes the
   earlier negotiated MSRP related data channel and both offerer and
   answerer may start using new the MSRP related data channel.

7. Security Considerations

   No security considerations are envisaged beyond those already
   documented in [RFC4566]

8. IANA Considerations

   To be completed.

9. Acknowledgments

   The authors wish to acknowledge the borrowing of ideas from other
   internet drafts by Salvatore Loreto, Gonzalo Camarillo, Peter Dunkley
   and Gavin Llewellyn, and to thank Paul Kyzivat, Jonathan Lennox, and
   Uwe Rauschenbach for their invaluable comments.

10. CHANGE LOG

10.1. Changes against '-01'

   o Formal syntax for dcmap and dcsa attribute lines.

   o Making subprotocol as an optional parameter in dcmap.

   o Specifying disallowed parameter combinations for max-time and max-
      retr.

   o Clarifications on data channel close procedures.

10.2. Changes against '-00'

   o Revisions to identify difference between internal and external
      negotiation and their usage.

   o Introduction of more generic terminology, e.g. "application"
      instead of "browser".

   o Clarification of how "max-retr and max-time affect the usage of
      unreliable and reliable data channels.

   o Updates of examples to take into account the SDP syntax changes
      introduced with draft-ietf-mmusic-sctp-sdp-07.

   o Removal of the SCTP port number from the a=dcmap and a=dcsa
      attributes as this is now contained in the a=sctp-port attribute,
      and as draft-ietf-mmusic-sctp-sdp-07 supports only one SCTP
      association on top of the DTLS connection.

11. References

11.1. Normative References

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

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

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

   [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax
              Specifications: ABNF", STD 68, RFC 5234, January 2008.

   [I-D.ietf-rtcweb-jsep]
              Uberti, J., Jennings, C., and E. Rescorla, "Javascript
              Session Establishment Protocol", draft-ietf-rtcweb-jsep-07
              (work in progress), July 2014.

   [I-D.ietf-rtcweb-data-channel]
              Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data
              Channels", draft-ietf-rtcweb-data-channel-12 (work in
              progress), September 2014.

   [I-D.ietf-mmusic-sctp-sdp]
              Loreto, S. and G. Camarillo, "Stream Control Transmission
              Protocol (SCTP)-Based Media Transport in the Session
              Description Protocol (SDP)", draft-ietf-mmusic-sctp-sdp-07
              (work in progress), July 2014.

   [WebRtcAPI]
              Bergkvist, A., Burnett, D., Jennings, C., and A.
              Narayanan, "WebRTC 1.0: Real-time Communication Between
              Browsers", World Wide Web Consortium WD-webrtc-20130910,
              September 2013,
              <http://www.w3.org/TR/2013/WD-webrtc-20130910/>.

11.2. Informative References

   [I-D.ietf-rtcweb-data-protocol]
              Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channel
              Establishment Protocol", draft-ietf-rtcweb-data-
              protocol-08 (work in progress), September 2014.

   [RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message
              Session Relay Protocol (MSRP)", RFC 4975, September 2007.

   [RFC4976] Jennings, C., Mahy, R., and A. Roach, "Relay Extensions
              for the Message Sessions Relay Protocol (MSRP)", RFC 4976,
              September 2007.

   [RFC5547] Garcia-Martin, M., Isomaki, M., Camarillo, G., Loreto, S.,
              and P. Kyzivat, "A Session Description Protocol (SDP)
              Offer/Answer Mechanism to Enable File Transfer", RFC 5547,
              May 2009.

   [RFC6135]   Holmberg, C. and S. Blau, "An Alternative Connection Model
               for the Message Session Relay Protocol (MSRP)", RFC 6135,
               February 2011.

   [RFC6714]   Holmberg, C., Blau, S., and E. Burger, "Connection
               Establishment for Media Anchoring (CEMA) for the Message
               Session Relay Protocol (MSRP)", RFC 6714, August 2012.

Authors' Addresses

   Keith Drage (editor)
   Alcatel-Lucent
   Quadrant, Stonehill Green, Westlea
   Swindon
   UK

   Email: keith.drage@alcatel-lucent.com

   Raju Makaraju
   Alcatel-Lucent
   2000 Lucent Lane
   Naperville, Illinois
   US

   Email: Raju.Makaraju@alcatel-lucent.com

   Juergen Stoetzer-Bradler
   Alcatel-Lucent
   Lorenzstrasse 10
   D-70435 Stuttgart
   Germany

   Email: Juergen.Stoetzer-Bradler@alcatel-lucent.com

   Richard Ejzak
   Unaffiliated

   Email: richard.ejzak@gmail.com

   Jerome Marcon
   Unaffiliated