Expressing encoder limits in CLUE

IETF88

Encoding limits in SDP

- Separate sendonly lines for each encoding
  - Allows expression of encoder send limits
- Necessitates additional O/A(s) for far end to add their own encodings

Initial O/A (A->B)

Offer (from A)

m=video ...

a=sendrecv

Answer (from B)

m=video ...

a=sendrecv

2nd O/A (A->B)

Offer (from A)

m=video ...
a=sendrecv
m=video ...
a=sendonly
m=video ...
a=sendonly
m=video ...
a=sendonly

Answer (from B)

m=video ...
a=sendrecv
m=video ...
a=recvonly
m=video ...
a=recvonly
m=video ...
a=recvonly

3rd O/A (B->A)

Offer (from B)

m=video ...
a=sendrecv
m=video ...
a=recvonly
m=video ...
a=recvonly
m=video ...
a=recvonly
m=video ...
a=sendonly
m=video ...
a=sendonly
m=video ...
a=sendonly

Answer (from A)

m=video ...
a=sendrecv
m=video ...
a=sendonly
m=video ...
a=sendonly
m=video ...
a=sendonly
m=video ...
a=recvonly
m=video ...
a=recvonly
m=video ...
a=recvonly

Encoding limits in CLUE

- Express encoding limits in ADVERTISEMENT message
  - Still have an m-line per encoding
- No longer need m-lines to be ‘sendonly’
- Express receive limits in SDP as normal

Offer SDP and Advertisment

Offer SDP

m=video ...
... H264@720p30
a=label:A
a=sendrecv
m=video ...
... H264@720p30
a=label:B
a=sendrecv
m=video ...
... H264@720p30
a=label:C
a=sendrecv

CLUE ADVERTISEMENT

Capture Scene 1:
Capture 1: Left (Encoding Group 1)
Capture 2: Center (Encoding Group 1)
Capture 3: Right (Encoding Group 1)
Capture 4: Switched
Capture Scene Entry 1: 1,2,3
Capture Scene Entry 2: 4
Simultaneous Sets: 1,2,3,4
Encoding Group 1:
Encoding 1: H264, 1080p30, label=A
Encoding 2: H264, 1080p30, label=B
Encoding 3: H264, 1080p30, label=C

Answer SDP and Configure

Answer SDP

m=video ...
...H264@720p30
a=sendrecv
m=video ...
... 264@720p30
a=sendrecv
m=video ...
... 264@720p30
a=sendrecv

CLUE CONFIGURE

Capture 1, Encoding 1
Capture 2, Encoding 2
Capture 3, Encoding 3

Advantages/Disadvantages

- Can use sendrecv for many m-lines, reduces number of O/As needed
- Less need to commit resources (ICE, etc) to encodings that aren't used
- Can express full send limitations

- Need to reinvent codec-specific language
  - This will be a large and neverending pain
- Or, we need suitably abstract language to not be codec-specific

Motivation: Why do we need individual encode limits in CLUE?

- Most normal SIP calls are OK with decoder limits only

- CLUE calls often have much more asymmetric media flows
- Receiver wants to make optimal decision on:
  - What to ask for
  - How to subdivide its decode resources

Motivation: Optimal decisions

If Alice offers 4 encodings, is that:

- 1 x 1080p
- 3 x 360p

OR

- 4 x 720p

Also reduces bandwidth and decoder resources allocated unnecessarily

What level of detail is required?

- Expressing individual encoder limits is done only to let receiver make better decisions on how to allocate resources
- Purely informative, and does not need to represent the reservation of actual resources
- As such, are generalities sufficient?