Session Announcement
(SAP, RFC 2974)

Session Description
(SDP, RFC 2327)
(SDP, draft-ietf-mmusic-sdp-new-25)

Slide contributions by Dirk Kutscher (Uni Bremen TZI)
IETF Multimedia (Conferencing) Architecture

- Resource Control
- Audio Video
- Session Directory
- Streaming
- Telephony
- Conferencing

- RTP
- SAP
- FLUTE
- RTSP
- SIP
- HTTP
- BFCP

- UDP
- SCTP
- TCP / TLS

IP / IP Multicast
Integrated / Differentiated Services Forwarding

Session Announcement Protocol (SAP)

- Announcing multimedia sessions to a broad audience
- Session announcements contain SDP
  - Subject of the session
  - Date(s) and time(s)
  - Media streams and addresses
  - Further information
- SAP Functions
  - New session announcements
  - Modify announcements
  - Delete announcements
  - Support for relays
- Earlier: Coordinate use of multicast address space
Dissemination of SAP Announcements

- Scope of Announcements
  - Per (administratively defined) multicast address scope
  - Local: 239.255.0.0/16
  - Organization local: 239.192.0.0/14
  - SAP conferences: 224.2.0.0 – 224.2.127.253
  - Other: Global
  - Similar considerations for IPv6
    - Scope identifier built-in into the IPv6 address structure

- SDP descriptions should use addresses of same scope
  - To ensure that receivers can also receive the media streams if they can receive the announcements
SAP Features

- **Limited announcement bandwidth per scope**
  - e.g. 4000 bit/s (defined per scope)

- **Calculation algorithm roughly similar to RTCP**
  - Measure incoming SAP packets per scope
    - Sizes, number of announcements
  - Calculate size of own announcements
  - Estimate available share of bandwidth
  - Calculate own transmission interval
    - Use dithering (± 1/3 of the interval)
    - Timer reconsideration before transmitting

(New) Announcements

- **SAP uses UDP/IP: no reliability**
- **Repeat announcements in “regular” intervals**
  - Intervals: in the order of minutes
    - e.g. minimum 5 min
- **Announcements for easy comparison identified by**
  - Source IP address (of the creator)
  - 16 bit hash value
- **May be authenticated (creator authentication)**
- **May be encrypted**
- **May be compressed**
- **May contain different payload types (SDP is just one)**
SAP Packet Format

- **V**: Version — =1 for SAPv2
- **A**: Address type — =0 IPv4 source address  
  =1 IPv6 source address
- **T**: Type — =0 Announcement packet  
  =1 Deletion packet
- **E**: Encrypted — indicates encryption of the announcement packet
- **C**: Compressed — indicates that the announcement packet is compressed
- **Auth Length** — Length of the authentication header (0 = no authentication)
SAP Header Fields (2)

- **Message ID Hash**: Unique value per session creator
- **Originating Source**: IP address of session creator
- **Authentication Data**: Source Authentication information (PGP and CMS formats defined so far)
- **Payload MIME Type**: NUL-terminated text string indicating the MIME type of the payload
  **Default**: application/sdp

Delete Announcements

- **Explicit Timeout**
  - No need to announce sessions after the “end time” in SDP
  - Caveat: the SAP receivers and relays need to understand SDP

- **Implicit Timeout**
  - Receiver observe repetition of announcement
  - After 10 times the announcement interval (or one hours) with re-announcement the session is removed

- **Explicit Deletion**
  - Send Deletion packet for a session
  - Message ID Hash and Originating Source must match
  - SHOULD be authenticated (match the original announcement)
Modify Announcements

- Replace an existing session description
  - E.g. modify media or start / end times
  - Update description
- Message ID Hash MUST change
- Modifying announcement MUST be authenticated if and only if the original announcement was
- If in doubt, a new session is “created”
  - Prevent denial-of-service attacks
- If proper match is found, the old session information is simply replaced by the new one

SAP Security

- Encrypted messages for secure information distribution
  - Should be only used with limited size receiver groups
  - Avoid waste of computation resource if many receivers cannot decrypt the message
  - Key distribution out-of-scope
  - Limited applicability, limited usefulness
- Authentication
  - SHOULD always be done
  - Enables at least to verify that two messages are from the same source
  - Proper source authentication requires PKI
- General observation
  - Both is rarely used in practice
  - Current use of SAP in the Internet does not justify the effort...
Session Announcement Tool: SDR

**Session Description Protocol (SDP)**

- All you need to know about a session to join
  - who? — convener of the session + contact information
  - what about? — name and informal subject description
  - when? — date and time
  - where? — multicast addresses, port numbers
  - which media? — capability requirements
  - how much? — required bandwidth
- Grouped into three categories
  - 1 x session, m x time, n x media
Session Level Description

- **v=0** Version
- **o=** Owner / creator of the session + unique identifier + version
- **u=** URL for further information
- **e=** Contact email address
- **p=** Contact phone number
- **b=** Bandwidth information
- **k=** Encryption key information
- **z=** Time zone adjustment
- **a=** Attribute lines (for extensions)
- **c=** Connection (=address) information

Time Description

- Start and end time(s) of a session
  - Plus time zone adjustment
- Regular repetitions
  - Every Tuesday and Thursday, 10 – 12
  - Every day
- Arbitrary repetitions
  - Repeated specification of t= lines

- **t=** Start, end time (NTP seconds, special case: 0, 0)
- **r=** Repetitions (interval, duration, offsets)
### Media Description

- Define the media streams comprising a conference
  - Media type (audio, video, text, tones, application, ...)
    - Only audio, video, text, tones are well-defined
  - (multicast) address(es) + port number
  - Maps RTP payload types for media to encoding formats
  - Other media level attributes

- **m** = Media and port specification
- **c** = IP address specification (inherited from session)
- **a** = Attributes for this media stream

### SDP Example

**Length of Time represented by Media in a single Packet**

<table>
<thead>
<tr>
<th>Session Information</th>
<th>Media Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=0</td>
<td>v=0</td>
</tr>
<tr>
<td>o=lynch 3117798688 3117798739 IN IP4 128.223.214.23</td>
<td>o=0</td>
</tr>
<tr>
<td>s=UO Presents KWAX Classical Radio</td>
<td>s=0</td>
</tr>
<tr>
<td>i=University of Oregon sponsored classical radio station KWAX-FM</td>
<td>i=0</td>
</tr>
<tr>
<td>u=<a href="http://darkwing.uoregon.edu/~uocomm/">http://darkwing.uoregon.edu/~uocomm/</a></td>
<td>u=0</td>
</tr>
<tr>
<td>e=UO Multicasters <a href="mailto:multicast@lists.uoregon.edu">multicast@lists.uoregon.edu</a></td>
<td>e=0</td>
</tr>
<tr>
<td>t=0 0</td>
<td>t=0 0</td>
</tr>
<tr>
<td>a=tool:sdr v2.4a6</td>
<td>a=tool:sdr v2.4a6</td>
</tr>
<tr>
<td>a=type:test</td>
<td>a=type:test</td>
</tr>
<tr>
<td>m=audio 30554 RTP/AVP 0</td>
<td>m=audio 30554 RTP/AVP 0</td>
</tr>
<tr>
<td>c=IN IP4 224.2.246.13/12</td>
<td>c=IN IP4 224.2.246.13/12</td>
</tr>
<tr>
<td>a=ptime:40</td>
<td>a=ptime:40</td>
</tr>
</tbody>
</table>
Session Description and Capability Negotiation

From Session Announcement to Session Invitation

Characteristics of SAP Announcements

- Common view
  - Every SAP-receiver sees the same description
    - Session meta information & scheduling
    - Media description & transport parameters

- Identical transport parameters for all participants
  - IP-Multicast service model:
    - Senders send to a multicast group (IP address)
    - Receivers join ("tune into") a multicast group
Session Initiation

- Distribute conference configuration
  - Applications
    - Media types, media format parameters
  - Transport Parameters
    - IP addresses, transport protocols, protocol parameters

- Negotiate Parameters!
  - Heterogeneous end systems
    - Different hardware and software capabilities
  - User preferences

- SDP provides syntax mechanisms to express parameters
  - Procedural model for initiation required

Invitation: Conceptual Model

INVITATION:
- List of applications
- List of supported configurations

RESPONSE:
- List of applications and configurations that are supported by A and B
- Selected Configuration and A’s Transport Parameters
- B’s Transport Parameters

Match A’s configuration with B’s configuration
Determine B’s transport parameters

Select one or more configurations, determine A’s transport parameters
Session Initiation with SDP (1)

A

```
v=0
o=s:alice 2890844526 2890844526 IN IP4 host.anywhere.com
s=phone call
h=EN IP4 host.anywhere.com
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

OK, send PCMU-Audio to host.anywhere.com/49170

B

```
v=0
o=bob 2890844730 2890844730 IN IP4 host.example.com
s=phone call
h=IN IP4 host.example.com
a=rtpmap:0 PCMU/8000
```

Send PCMU-Audio to host.example.com/49920

Session Initiation with SDP (2)

A

```
v=0
o=s:alice 2890844526 2890844526 IN IP4 host.anywhere.com
s=phone call
h=IN IP4 host.anywhere.com
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 51372 RTP/AVP 31
a=rtpmap:31 H261/90000
```

Send PCMU-Audio to host.anywhere.com/49170 and H261-Video to host.anywhere.com/51372

B

```
v=0
o: bob 2890844730 2890844730 IN IP4 host.example.com
s=phone call
h=IN IP4 host.example.com
m=audio 49920 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 0 RTP/AVP 31
```

OK, send PCMU-Audio to host.example.com/49920. I cannot or do not want to receive video.
SDP Offer/Answer Model (RFC 3264)

- For initiation of unicast sessions
- Objective: generate common view of session configuration
- Simple exchange of capability descriptions

- Basic Model:
  - A sends offer to B, including
    - Set of media streams and codecs A wishes to use
    - Transport parameters (where A wants to receive data)
  - B sends answer to A
    - For each stream in offer, indicating whether stream is accepted or not
    - For each stream add transport parameters (where B wants to receive data)

Codec Selection

Offerer

A

B may send PCMU, G721 or GSM to A.
A must be prepared for B changing codecs dynamically.

I am prepared to receive PCMU, G721 or GSM encoded audio.

Answerer

B

I can only receive PCMU encoded audio.
Codec Selection

- Offer can provide multiple codecs for a media stream.
  - Ordered by preference
  - Offerer commits to support all codecs (one at a time)
  - Answerer should generate list of codecs for each stream, maintaining payload type mapping
  - New codecs may be added

- One of N codec selection
  - Offer multiple codecs, but cannot change dynamically
  - Offerer sends codec list "with reservation"
  - Answerer sends back subset
  - Offerer "locks" one codec for session
  - Implemented with a=inactive media level attribute...

Unidirectional Streams

Offerer

A

Offerer Answerer

I only want to receive PCMU-Audio. Please send to host.anywhere.com/49170.

Answerer

B

OK, I will send PCMU audio to you. BTW, for this RTP stream I expect RTCP feedback on port 49920+1 (at host.example.com).
Unidirectional Streams

- a=sendonly, a=recvonly, a=inactive

Grouping of m= lines in SDP

- Observation:
  - Multiple m= lines in SDP have no relationship to each other
    - Independent media streams
    - usually different media types

- Problem:
  - Want to express synchronization relationship
    - Lip synchronization
  - Concept of “flows” that consist of several media streams
    - Streams encoded in several formats
    - May be streamed from different hosts/ports
    - Useful application in some IP telephony scenarios
Example for Lip Synchronization

Stream 1 and 2 should be synchronized.

v=0
o=Laura 289083124 289083124 IN IP4 one.example.com
t=0 0
c=IN IP4 224.2.17.12/127
a=group:LS 1 2
m=audio 30000 RTP/AVP 0
a=mid:1
m=video 30002 RTP/AVP 31
a=mid:2
i=This media stream contains the Spanish translation

ANAT Grouping

- Alternative Network Address Types (RFC 4091)
  - Allows expressing IPv4 and IPv6 address alternatives

v=0
o=bob 280744730 28977631 IN IP4 host.example.com
s=
t=0 0
a=group:ANAT 1 2
m=audio 25000 RTP/AVP 0
c=IN IP6 2001:DB8::1
a=mid:1
M=audio 22334 RTP/AVP 0
c=IN IP4 192.0.2.1
a=mid:2
FEC Grouping

- Group basic and FEC data (draft-ietf-mmusic-fec-grouping-00.txt)

\[
v=0  
c=adam 289083124 289083124 IN IP4 host.example.com  
s=ULP FEC Seminar  
t=0 0  
c=IN IP4 224.2.17.12/127  
a=group:FEC 1 2  
a=group:FEC 3 4  
m=audio 30000 RTP/AVP 0  
amid:1  
m=application 30002 RTP/AVP 100  
amid:2  
m=video 30004 RTP/AVP 31  
amid:3  
m=application 30004 RTP/AVP 101  
c=IN IP4 224.2.17.13/127  
amid:4  
\]

Further Groupings

- General considerations so far only

- Alternative RTP profiles
  - E.g. AVP and AVPF, AVP and SAVP

- Alternative addresses
  - Multicast vs. unicast distribution
  - E.g. for a media streaming offer
Simple Capability Declaration in SDP

Observation:
- Capability negotiation/declaration in SDP too limited
- Session description describe both session parameters and capabilities without clear distinction
- Simultaneous capability restrictions cannot be expressed
  - "Supporting multiple codecs for one media type, but only one per session"

Simcap: add SDP attributes to explicitly express capabilities

Simcap Example

Sender is willing to receive and send G.729 (18) and telephone-events.

Additionally, it declares the following capabilities:
- PCMU-Audio (0)
- telephone-events (different events)
- Fax-Relay over UDP and TCP

v=0
o=- 25678 753849 IN IP4 128.96.41.1
s=
c=IP4 128.96.41.1
m=audio 3456 RTP/AVP 18 96
a=rtpmap:96 telephone-event
a=fmtp:96 0-16,32-35
a=cdsc: 1 audio RTP/AVP 0 18 96
a=cpar: a=fmtp:96 0-16,32-35
a=cdsc: 4 image udptl t38
a=cdsc: 5 image tcp t38
Simcap Example

Semantics:

- `a=sqn`: declares a sequence number
- `a=cldsc`: declare one or more capabilities
- `a=cpnr`: additional parameters for a declaration

Connection-oriented Media with SDP

- Focus on TCP (RFC 4145)
  - TLS in progress
- In contrast to UDP, a connection must be established
  - Who is to initiate setup, who is to listen?
    - `a=setup: active | passive | actpass | holdconn`
  - What if a connection already exists (e.g., when renegotiating)
    - Keep the existing connection?
    - Set up a new one?
    - `a=connection: new | existing`
  - When to tear down a connection?
    - If a "new" one is specified, close an existing one
- Relies on interactive agreement on how to proceed
Labeling media streams

- **Unique identification**
  - Across SDP session descriptions
    - Contrast to mid (which is valid within a session only)
  - `a=label:<token>`
  - No semantics

- **Attaching stream semantics**
  - Usually relevant within an SDP session
  - Hint at stream semantics
    - E.g., if multiple media streams are received: which is which?
  - `a=content:<token>`
  - `token=slides | speaker | sl | main | alt | user-floor | ...`

---

Session Management Attributes

- **Signaling the RTCP port (RFC 3605)**
  - Motivation: RTP and RTCP port number may not be adjacent
  - `a=rtcp:<port> [ <nettype> <addrtype> <addr> ]`
  - `a=rtcp:60004 [ IN IP4 192.168.11.12 ]`

- **Signaling multicast sources (IMGPv3, SSM)**
  - `a=src-filter:incl IN IP4 232.3.4.5 192.168.1.89`
  - `a=src-filter:excl IN IP4 225.3.4.5 192.168.1.89 192.168.6.66`

- **Session bandwidth (independent of lower layers, RFC 3890)**
  - `b=TIAS:64000`
  - `a=maxprate:40.0`

- **RTCP bandwidth (modify sender/receiver share, RFC3556)**
  - `b=RS:1600`
  - `b=BR:14400`
SDP Extensions: There is more…

- Precondition signaling for media streams
  - Security
  - QoS
  - Connectivity
- Key management (fixing k=)
  - End-to-end key negotiation
  - End-to-end key distribution (via a protected channel)
- And support for further media types
  - Multicast file distribution, application sharing, ...

Summary So Far

- SDP syntax can be used for session initiation
  - But requires additional specification of procedures: Offer/Answer
- SDP & Offer/Answer not appropriate for all usage scenarios
  - Fundamental SDP problem of combining configuration descriptions with capability declaration
  - Lack of expressiveness: grouping of media streams
  - “a=” only a limited extension mechanism
SDP Syntax Issues

- **Basic** set of description elements for media sessions
  - IP addresses, port numbers, RTP payload types, parameters

- **Extensibility:** new session / media level attributes
  - `a=<keyword>:<value> ...`
  - Senders can use arbitrary attributes:
    - Important attributes cannot be distinguished from unimportant ones
    - Name clashes (misinterpretation) cannot be excluded
  - In principle, allows for any kind of extension
    - Grouping, constraints, ...

- SDP workarounds rather clumsy, inefficient, ...

SDP Next Generation (SDPng)

- Being designed to address SDP’s flaws...
  - Limited expressiveness
    - For individual media and their combination
    - Often only very basic media descriptions available
  - No real negotiation functionality
  - Limited extensibility (clumsy, hard to coordinate)
  - No semantics for media sessions (only implicit)

- Also: Avoid second system syndrome!
  - Simple, easy to parse, extensible, limited scope
Intelligent Endpoints

- Intelligent endpoints with support for
  - Multiple codecs and format parameters
  - Different applications (e.g., audio, DTMF, video, games)
  - Many transport parameters
    - RTP/UDP/IPv4, RTP/UDP/IPv6, Security, Source-Specific-Multicast...
  - AAA & security parameters

Must be expressible in configuration descriptions!

Intelligent Endpoints

- Heterogeneous end systems
  - Different capabilities
  - Different user preferences
  - Dynamic configuration

Interoperability requires dynamic negotiations of parameters!
Specific Requirements

- **Expressiveness**
  - Describe all required configuration parameters

- **Extensibility**
  - No fixed parameter set
  - Profiles ("packages") for new configuration parameters

- **Support for Negotiation**
  - Derive commonly supported configurations from individual configuration descriptions (for $n \geq 2$)

- **Compatibility**
  - Drop-in replacement for SDP in SIP applications

SDPng’s Conference Model

- **Components in a conference**
  - Individual cooperation functions
  - Characterized by the service they provide (not by their technical implementation)

- **Implementations of components**
  - Depend on endpoint capabilities and user preferences
  - Use of implementations must be configured or negotiated
Potential Configurations

- Configurations for implementing a component
  - Common capabilities
  - Not a complete conference description, e.g., no transport parameters
  - Dynamic set of parameters
    - Can change over the course of a conference

Actual Configurations

- Complete specification of conference parameters
  - Selected subset of potential configurations
  - Complemented with
    - Media format parameters
    - Transport parameters

```
address=192.168.1.1
port=37000
codec-type=PCMU
payload-type=0
...
```
General SDPng Model

Potential Configurations

Definitions

Actual Configurations

Constraints

Session-Level Info

SDP m= blocks without transport parameters

“optional”

SDP m= blocks

“optional”

SDP session attributes + stream semantics

SDPng Structure

Potential Configurations

List of capabilities as XML elements. Only these are are processed by capability negotiation.

Definitions

Define commonly used parameters for later referencing.

Actual Configurations

Actual configurations as alternatives for each component.

Constraints

Reference configurations and express constraints on combinations

Session-Level Info

Elements for meta information on individual applications (i.e., streams, sessions), referencing configuration definitions.
SPDng: An Extensible Framework

SPDng consists of
- **Base specification**
  - Overall structure of SPDng documents
  - Common data types and element types
- **Basic rules packages (“profiles”)**
  - Define how to express commonly used parameters
    - Codecs, RTP parameters etc.
- **Basic definitions (“libraries”)**
  - Specific codec definitions, RTP payload type definitions etc.

Formally specified

SDPng description instances

Capability Model

- **Three different types**
  - Tokens:
    - `encoding=PCMU`
    - `Ascertain identity || fail`
  - Token lists:
    - `sampling-rate=8000,16000, 44000`
    - `Determine common subset || fail`
  - **Numerical Ranges**
    - `6 <= bitrate <= 64`
    - `Determine common sub-range || fail`
- **Distinguish optional capabilities**
  - `silence-suppression supported`
  - Applicable to each type, failing results in removing the capability, interoperability still possible
XML Syntax (1)

- Feature independent negotiation
  - Process capability descriptions without knowing semantics
  - Access to schema definition not required

XML Syntax (2)

- Capabilities
  - A collection of independent definitions
  - Each definition is processed independently
  - Every property is a single XML element
    - Tokens and token lists as element content
    - Numerical ranges with explicit XML attributes
    - No further substructure
    - Descriptions are still standalone

```xml
<audio:codec name="avp:pcmu">
  <audio:encoding>PCMU</audio:encoding>
  <audio:channels>1 2</audio:channels>
  <audio:sampling>8000 16000</audio:sampling>
  <audio:bitrate min="6" max="64"/>
  <audio:silence-suppression status="opt"/>
</audio:codec>
```
Formal Schema Definition

- Base specification
  - SDPng XML document structure
  - Basic data types (token, token lists, ranges)
  - XML-Schema as a definition mechanism

- Package definitions
  - Application specific vocabulary
  - Each package definition in unique XML namespace
  - XML-Schema as a definition mechanism

Sample Package Definition

```xml
<xsd:complexType name="audio:CodecT">
  <xsd:complexContent>
    <xsd:extension base="sdpng:Definition">
      <xsd:sequence>
        <xsd:element name="encoding" type="sdpng:token"/>
        <xsd:element minOccurs="0" name="channels" type="sdpng:tokenlist"/>
        <xsd:element minOccurs="0" name="sampling" type="sdpng:tokenlist"/>
        <xsd:element minOccurs="0" name="bitrate" type="sdpng:range"/>
        <xsd:element minOccurs="0" name="silenceSuppression" type="sdpng:optToken"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="audio:codec" type="audio:CodecT" substitutionGroup="sdpng:definition"/>
```
Specifying Configurations (1)

```
<cap>
  <audio:codec name="avp:pcmu">
    <audio:encoding>PCMU</audio:encoding>
    <audio:channels>1 2</audio:channels>
    <audio:sampling>8000 16000</audio:sampling>
    <audio:bitrate min="6" max="64"/>
    <audio:silence-suppression status="opt"/>
  </audio:codec>
  <rtp:udp name="rtpudpip6">
    <rtp:network>IP6</rtp:network>
  </rtp:udp>
</cap>
```

Specifying Configurations (2)

```
<cap>
  <audio:codec name="avp:pcmu"> […] </audio:codec>
  <rtp:udp name="rtpudpip6"> […] </rtp:udp>
</cap>

<def>
  <rtp:udp name="rtp-cfg1" ref="rtp:rtpudpip6">
    <rtp:ip-addr>::1</rtp:ip-addr>
    <rtp:port>9456</rtp:port>
    <rtp:pt>1</rtp:pt>
  </rtp:udp>
</def>
```
Specifying Configurations (3)

```
<cap>
  <audio:codec name="avp:pcmu"> [...] </audio:codec>
  <rtp:udp name="rtpudpip6"> [...] </rtp:udp>
</cap>

<def>
  <rtp:udp name="rtp-cfg1"> [...] </rtp:udp>
</def>

<cfg>
  <component name="interactive-audio" media="audio">
    <alt name="alt1">
      <audio:codec ref="avp:pcmu"/>
      <rtp:udp ref="rtp-cfg1"/>
    </alt>
  </component>
</cfg>
```

Specifying Configurations (4)

- Each component (application session) element provides list of alternatives
- Each alternative provides definitions for the component
  - Referencing definitions from the capability section
    - Providing additional parameters, where required
    - Alternatives that reference non-interoperable definitions are discarded
  - List of definitions
    - No nesting of elements from different packages
  - Semantics are application-specific
    - Applications MUST know how to interpret definitions
  - No restrictions on quantity or order
Libraries

- Libraries:
  - Pre-defined definitions, e.g., a set of audio codec definitions
  - Referenced from a description document

- Semantics difficult to get right
  - Application-independent negotiation would require access to library definitions
    - Requirement to *include* library definitions into description document
    - Capability negotiation has to consider *all* definitions

- Forego libraries, include definitions inline

Summary

- Extensibility and dynamic negotiation are key to interoperability
  - Intelligent endpoints and new services require a capable and flexible description mechanism

- SDPng to provide interoperability *and* extensibility
  - Simple applications stay simple
  - Innovation is possible through structured extensibility

- Smooth migration from SDP to SDPng is possible
  - “Bi-lingual” endpoints and mapping of SDP to SDPng