Kutscher
Ott
Bormann
TZI, Universitaet Bremen
November 24, 2000

# Requirements for Session Description and Capability Negotiation
draft-kutscher-mmusic-sdpng-req-01.txt

## Status of this Memo



## Copyright Notice



## Abstract

This document defines some terminology and lists a set of requirements that are relevant for a framework for session description and endpoint capability negotiation in multiparty multimedia conferencing scenarios.

This document is intended for discussion in the Multiparty Multimedia Session Control (MMUSIC) working group of the Internet Engineering Task Force. Comments are solicited and should be addressed to the working group's mailing list at confctrl@isi.edu and/or the authors.

Table of Contents

# 1. Introduction

Multiparty multimedia conferencing is one application that requires the dynamic interchange of end system capabilities and the negotiation of a parameter set that is appropriate for all sending and receiving end systems in a conference. For some applications, e.g. for loosely coupled conferences, it may be sufficient to simply have session parameters be fixed by the initiator of a conference. In such a scenario no negotiation is required because only those participants with media tools that support the predefined settings can join a media session and/or a conference.

This approach is applicable for conferences that are announced some time ahead of the actual start date of the conference. Potential participants can check the availability of media tools in advance and tools like session directories can configure media tools on startup. This procedure however fails to work for conferences initiated spontaneously like Internet phone calls or ad-hoc multiparty conferences. Fixed settings for parameters like media types, their encoding etc. can easily inhibit the initiation of conferences, for example in situations where a caller insists on a fixed audio encoding that is not available at the callee's end system.

To allow for spontaneous conferences, the process of defining a conference's parameter set must therefore be performed either at conference start (for closed conferences) or maybe (potentially) even repeatedly every time a new participant joins an active conference. The latter approach may not be appropriate for every type of conference without applying certain policies: For conferences with TV-broadcast or lecture characteristics (one main active source) it is usually not desired to re-negotiate parameters every time a new participant with an exotic configuration joins because it may inconvenience existing participants or even exclude the main source from media sessions. But conferences with equal ``rights'' for participants that are open for new participants on the other hand would need a different model of dynamic capability negotiation, for example a telephone call that is extended to a 3-parties conference at some time during the session.

SDP [1] allows to specify multimedia sessions (i.e. conferences, "session" as used here is not to be confused with "RTP session"!) by providing general information about the session as a whole and specifications for all the media streams (RTP sessions and others) to be used to exchange information within the multimedia session.

Currently, media descriptions in SDP are used for two purposes:

o to describe session parameters for announcements and invitations

(the original purpose of SDP)

- to describe the capabilities of a system (and possibly provide a choice between a number of alternatives). Note that SDP was not designed to facilitate this.

A distinction between these two "sets of semantics" is only made implicitly.

In the following we first introduce a model for session description and capability negotiation and define some terms that are later used to express some requirements. Note that this list of requirements is possibly incomplete. The purpose of this document is to initiate the development of a session description and capability negotiation framework.

## 2. Terminology and System Model

Any (computer) system has, at a time, a number of rather fixed hardware as well as software resources. These resources ultimately define the limitations on what can be captured, displayed, rendered, replayed, etc. with this particular device. We term features enabled and restricted by these resources "system capabilities".

Example: System capabilities may include: a limitation of the screen resolution for true color by the graphics board; available audio hardware or software may offer only certain media encodings (e.g. G.711 and G.723.1 but not GSM); and CPU processing power and quality of implementation may constrain the possible video encoding algorithms.

In multiparty multimedia conferences, participants employ different "components" in conducting the conference.

Example: In lecture multicast conferences one component might be the voice transmission for the lecturer, another the transmission of video pictures showing the lecturer and the third the transmission of presentation material.

Depending on system capabilities, user preferences and other technical and political constraints, different configurations can be chosen to accomplish the ‘‘deployment’’ of these components.

Each component can be characterized at least by (a) its intended use (i.e. the function it shall provide) and (b) a one or more possible ways to realize this function. Each way of realizing a particular function is referred to as a "configuration".

Example: A conference component’s intended use may be to make transparencies of a presentation visible to the audience on the Mbone. This can be achieved either by a video camera capturing the image and transmitting a video stream via some video tool or by loading a copy of the slides into a distributed electronic whiteboard. For each of these cases, additional parameters may exist, variations of which lead to additional configurations (see below).

Two configurations are considered different regardless of whether they employ entirely different mechanisms and protocols (as in the previous example) or they choose the same and differ only in a single parameter.

Example: In case of video transmission, a JPEG-based still image protocol may be used, H.261 encoded CIF images could be sent as could H.261 encoded QCIF images. All three cases constitute different configurations. Of course there are many more detailed protocol parameters.

Each component's configurations are limited by the participating system's capabilities. In addition, the intended use of a component may constrain the possible configurations further to a subset suitable for the particular component's purpose.

Example: In a system for highly interactive audio communication the component responsible for audio may decide not to use the available G.723.1 audio codec to avoid the additional latency but only use G.711. This would be reflected in this component only showing configurations based upon G.711. Still, multiple configurations are possible, e.g. depending on the use of A-law or u-Law, packetization and redundancy parameters, etc.

In this system model, we distinguish two types of configurations:

o potential configurations
(a set of any number of configurations per component) indicating a system's functional capabilities as constrained by the intended use of the various components;

o actual configurations
(exactly one per instance of a component) reflecting the mode of operation of this component's particular instantiation.

Example: The potential configuration of the aforementioned video component may indicate support for JPEG, H.261/CIF, and H.261/QCIF. A particular instantiation for a video conference may use the actual configuration of H.261/CIF for exchanging video streams.

In summary, the key terms of this model are:

o A multimedia session (streaming or conference) consists of one or more conference components for multimedia "interaction".

o A component describes a particular type of interaction (e.g. audio conversation, slide presentation) that can be realized by means of different applications (possibly using different protocols).

o A configuration is a set of parameters that are required to implement a certain variation (realization) of a certain component. There are actual and potential configurations.

  * Potential configurations describe possible configurations that are supported by an end system.

* An actual configuration is an "instantiation" of one of the potential configurations, i.e. a decision how to realize a certain component.

In less abstract words, potential configurations describe what a system can do ("capabilities") and actual configurations describe how a system is configured to operate at a certain point in time (media stream spec).

To decide on a certain actual configuration, a negotiation process needs to take place between the involved peers:

1. to determine which potential configuration(s) they have in common, and

2. to select one of this shared set of common potential configurations to be used for information exchange (e.g. based upon preferences, external constraints, etc.).

In SAP [11] -based session announcements on the Mbone, for which SDP was originally developed, the negotiation procedure is non-existent. Instead, the announcement contains the media stream description sent out (i.e. the actual configurations) which implicitly describe what a receiver must understand to participate.

In point-to-point scenarios, the negotiation procedure is typically carried out implicitly: each party informs the other about what it can receive and the respective sender chooses from this set a configuration that it can transmit.

Capability negotiation must not only work for 2-party conferences but is also required for multi-party conferences. Especially for the latter case it is required that the process of determining the subset of allowable potential configurations is deterministic to reduce the number of required round trips before a session can be established.

In the following, we elaborate on requirements for an SDPng specification, subdivided into general requirements and requirements for session descriptions, potential and actual configurations as well as negotiation rules.

# 3. General Requirements

Note that the order in which these requirements are presented does not imply their relative importance.

## 3.1 Simplicity

The SDPng syntax shall be simple to parse and the protocol rules shall be easy to implement.

## 3.2 Extensibility

SDPng shall be extensible in a backward compatible fashion. Extensions should be doable without modifying the SDPng specification itself. The spec should preclude two independent extensions from clashing with each other (e.g. in the naming of attributes).

Along with extensibility comes the requirement to identify certain extensions as mandatory in a given context while others as optional.

## 3.3 Firewall Friendliness

It should be theoretically possible for firewalls (and other network infrastructure elements) to process announcements etc. that contain SDPng content. The concrete procedures have to be defined but if possible the processing of the SDPng content should be doable without interpretation of the textual descriptions.

## 3.4 Security

SDPng should allow independent security attributes for parts of a session description. In particular, signing and/or encrypting parts of a session description should be supported.

## 3.5 Text encoding

A concise text representation is desirable in order to enhance portability and allow for simple implementations. At run time, size of encoded packets should be minimized, processing as well.

A language that allows specifications to be formally validated is desirable.

A tendency to use XML as basis for the specification language has been expressed repeatedly.

## 3.6 Session vs. Media Description

In many application scenarios (particularly with SIP and MEGACO/H.248), only media descriptions are needed and there is no need for session description parameters. SDPng should make parameter sets optional where it is conceivable that not all application will need them.

## 3.7 Mapping (of a Subset) to SDP

It shall be possible to translate a subset of SDPng into standard SDP session description to enable a certain minimal degree of interoperability between SDP-based and SDPng-based systems. However, as SDPng will provide enhanced functionality compared to SDP, a full mapping to SDP is not possible.

Note: Backwards compatibility to the SDP syntax has been discussed and it was found that this is not goal for SDPng, as it is felt that RFC 2327 is too limiting.

Since several flavors of SDP have been developed (e.g., the MEGACO WG uses certain non-SDP enhancements) it needs to be discussed which of these flavors need to be considered for some kind of mapping.

## 4. Session Description Requirements

For now, we only consider requirements for media (stream) descriptions.

### 4.1 Media Description

It must be possible to express the following information with SDPng:

#### 4.1.1 Medium Type

Payload types and format parameters for audio and video are well-defined and the basic semantics are clear (as defined in RFC1889 [2] and RFC2327 [1]).

Format descriptions for text and whiteboard are currently only defined in the context of specific applications, this is probably going to change in the future (not an SDPng work item).

Non-standard (in terms of defined as a non-standard payload type) codecs and format parameters can be accomplished by using dynamic payload type mappings. This is a crucial feature of SDP that needs to be preserved for RTP applications.

Current SDP only provides a= (a=fmtp) as means to specify codec parameters but actually gives little support on how to do this. Schemes for expressing more sophisticated parameters (e.g. supporting nesting) may be necessary. Nevertheless, it is imperative to keep the overall structure of a codec description manageable.

Note that there is a conflict between the desire to be able to use any old SDP and translate it in SDPng and the desire to have a useful structure in the SDPng data.

#### 4.1.2 Media Stream Packetization

SDPng needs to be able to take care of more sophisticated payload descriptions than simple payload type assignment. Audio/video redundancy coding schemes need to be supported as need other mechanisms for FEC (RFC 2733 [7]) and media stream repair (RFC 2354 [8]). Also, layered coding schemes need to be supported.

Finally, a separation of the media encoding scheme, the packetization format, and possible repair schemes (and their respective parameters) is required.

#### 4.1.3 Transport

Since session descriptions are not only used to describe sessions that use IPv4/RTP for media transport it must be possible to specify different transport protocols (and their corresponding mandatory parameters). This means SDPng must support different address formats (IPv4, IPv6, E.164, NSAP, ...), multiplexing schemes (e.g. to identify a channel on a TDM link), and different transport protocol stacks (RTP/UDP/IP, RTP/AAL5/ATM, ...). Potential further parameters and interdependencies for multiplexed transports should be considered.

Additionally the requirement for expressing multiple addresses per actual configuration (layered coding support) has emerged, as well as the requirement for expressing multiple addresses per potential configuration (one port per payload type to simplify processing at the receiver). (A motivation has been provided by draft-camarillo-sip-sdp-00.txt [9].)

In multi-unicast-scenarios it must be possible to specify more than one transport address for a single media stream in an actual configuration, i.e. by specifying address lists.

In "broadcast"- or "lecture"-like sessions source filters might be needed that allow receivers to verify the source and apply filters in multicast sessions. Similarly, for SSM, the transport address includes an (Sender,Group) pair of IP addresses.

#### 4.1.4 QoS

QoS-Parameters for different protocol domains (e.g. traffic specification and flow specification or TOS bits for IP QoS) need to be specified. draft-ietf-mmusic-sdp-qos-00.txt [10] has provided a proposal for a syntax that can be used with SDP to describe network and security preconditions that have to be met in order to establish a session.

#### 4.1.5 Resource Utilization

A requirement debated (but not yet agreed upon) was whether abstract terms should be found to describe resource requirements (in terms of CPU cycles, DSPs, etc.)

#### 4.1.6 Dependencies

Certain codes may depend on other resources being available (e.g. a G.723.1 audio codec may need a DTMF codec as well while a G.711 codec does not). Such interdependencies need to be expressed.

#### 4.1.7 Other parameters (media-specific)

Extension mechanisms that allow to describe arbitrary other parameters of media codecs and formats are mandatory. It is possibly required to distinguish between mandatory and optional extension parameters.

In particular, it must be possible to introduce new (optional) parameters for a payload format and have old implementations still parse the parameters correctly.

#### 4.1.8 Naming Hierarchy and/or Scoping

Parameter names should be constructed in a way to avoid clashes and thereby simplify independent development of e.g. codec parameter descriptions in different groups.

# 5. Requirements for Capability Description and Negotiation

## 5.1 Capability Constraints

Capability negotiation is used to gain a session description (an actual configuration) that is compatible with the different end system capabilities and user preferences of the potential participants of a conference.

A media capability description is the same as a potential configuration, as it contains a set of allowable configurations for different components that could be used to implement the corresponding component. A capability description should allow specifying a number of interdependencies among capabilities. Traditional SDP only supports alternative capabilities and the specification implicitly assumed that all capabilities could be combined and basically used at the same time (looking at the pure session description, at least).

Processing power, hardware, link, or other resources may preclude the simultaneous use of certain configurations and/or limit the number of simultaneous instantiations of one or more configurations. This has led to a need to express in more detail constraints on combinations of configurations including the following constraints:

- o grouping capabilities (-> capability set);
- o expressing simultaneous capability sets;
- o expressing alternative capability sets; and
- o constraining the number of uses of a certain capability (set).

It needs to be carefully investigated how much more sophistication (if any) than simply listing alternatives needs to go into a base specification of SDPng (and which extension mechanisms for certain applications or for future revisions should be allowed).

Examples are known where complex capability descriptions are available but are simply not used (at least not at the level of sophistication that would be possible). This strongly calls for keeping requirements on capability constraints rather modest (KISS).

## 5.2 Processing Rules

The processing of potential configurations includes the process of "collapsing" sets of potential configurations offered by participants, i.e. the computation of the intersection of these potential configurations.

The processing (i.e. collapsing, forwarding etc.) of different potential configurations in order to find a compatible subset must work without having to know the semantics of the individual parameters. This is a key requirement for extensibility.

Additionally it must be possible to make use of different negotiation policies in order to reflect different conference types. For example in a lecture-style conference the policy might be to ensure that a capability collapsing process does not yield an actual configuration that excludes the main source (i.e. the lecturer and her end system) from the conference.

Preferences may also be considered in the negotiation process. This may need to be considered at the SDPng level (e.g. to express preferences, priorities).

Of course, the negotiation of configurations must not only work in peer-to-peer-conference scenarios but also be usable in multi party scenarios.

Negotiation of capabilities should take no longer than two or three message exchanges. The description format must enable such efficiency.

In order to allow for concise capability specification it will probably be required to group descriptions of, say, codecs and to establish a kind of hierarchy that allows to attach a certain attribute or parameter to a whole group of codecs.

It might then also be required to have a naming scheme that allows to name definitions in order to be able to later reference them in subsequent definitions. This is useful in situations where some definition extends a previous definition by just one parameter or in situations where codecs are combined, for example for expressing redundancy or layered codings. Different models of re-use are conceivable.

## 6. Remarks

Explicitly addressing the issue of capability negotiation when drafting the new session description language generates new sets of requirements, some of which might conflict with other important goals, such as simplicity, conciseness and SDP-compatibility.

However, we think that it's worthwhile to sketch a reasonably complete and powerful solution first and then later develop a migration path from today's technology instead of imposing limitations at the outset to minimize the possibly necessary changes.

## 7. SDPng: A Strawman Proposal

This section outlines a proposed solution for describing capabilities that meets most of the above requirements. Note that at this early point in time not all of the details are completely filled in; rather, the focus is on the concepts of such a capability description and negotiation language.

### 7.1 Conceptual Outline

Our concept for the description language follows the system model introduced in the beginning of this document. We use a rather abstract language to avoid misinterpretations due to different intuitive understanding of terms as far as possible.

PLEASE NOTE that the examples in the following are given for illustrative purposes only; they are not meant to be syntactically complete or consistent. For a more real example refer to the end of this section.

Our concept of a capability description language addresses various pieces of a full description of system and application capabilities in four separate "sections":

Definitions (elementary and compound)

Potential or Actual Configurations

Constraints

Session attributes

#### 7.1.1 Definitions

The definition section specifies a number of "entities" that are later referenced to avoid repetitions in more complex specifications and allow for a concise representation. Entities are identified by an "id" by which they may be referenced. Entities may be elementary or compound (i.e. combinations of elementary entities).

Elementary entities do not reference other entities. Each elementary entity only consists of one of more attributes and their values. Default values specified in the definition section may be overridden in descriptions for potential (and later actual) configurations.

For the moment, elementary entities are defined for media types (i.e. codecs) and for media transports. For each transport and for each codec to be used, the respective attributes need to be defined.

This definition may be either within the "Definition" section itself or in an external document (similar to the audio-visual profile or an IANA registry that define payload types and media stream identifiers.

Examples for elementary entities include "{media=audio, coding=PCM, compression=ulaw, rate=8000}" to be identified by id="PCMU" and "{transport=UDP, framing=RTP, network=IPv4, ...}" to be identified by id="AVP".

Compound entities combine a number of elementary and/or other compound entities for more complex descriptions. This mechanism can be used for simple standard configurations such as G.711 over RTP/AVP as well as to express more complex coding schemes including e.g. FEC schemes, redundancy coding, and layered coding. Again, such definitions may be standardized and externalized so that there is no need to repeat them in every specification.

An example for a redundant audio payload format (following RFC 2198) could be "{media=audio, coding=rfc2198, primary=ref:PCMU, secondary=ref:GSM, pattern=1:2, pt=97}" referred to by id="G711-Red". Standard uncompressed IP telephony audio could be "{transport=ref:AVP, codec=ref:PCMU}" identified by id="IPTEL-UNC".

Both types of entities may have default values specified along with them for each attribute. Some of these default values may be overridden so that a codec definition can easily be re-used in a different context (e.g. by specifying a different sampling rate) without the need for a large number of base specifications.

This approach taken here allows to have simple as well as more complex definitions which are commonly used be available in an extensible set of reference documents. Care should be taken though not to make the external references too complex and thus require too much a priori knowledge in a protocol engine implementing SDPng.

Note: For negotiation between endpoints, it may be helpful to define two modes of operation: explicit and implicit. Implicit specifications may refer to externally defined entities to minimize traffic volume, explicit specifications would list all external definitions used in a description in the "Definitions" section.

#### 7.1.2 Components & Configurations

The "Configurations" section contains all the components that constitute the multimedia conference, IP telephone call, etc. For each of these components, the potential and, later, the actual configurations are given. Potential configurations are used during capability exchange and/or negotiation, actual configurations to configure media streams after negotiation or in session announcements (e.g. via SAP). A potential and the actual configuration of a component may be identical.

Each component has an identifier ("id") so that it can be referred to, e.g. to associate semantics with a particular media stream. For such a component, any number of configurations may be given with each configuration describing an alternate way to realize the functionality of the respective component.

Each configuration (potential as well as actual) is identified by an "id". A configuration combines one or more (elementary and/or compound) entities from the "Definitions" section to describe a potential or an actual configuration. Within the specification of the configuration, default values from the referenced entities may be overwritten.

For example, an IP telephone call may require just a single component id=interactive-audio with two possible ways of implementing it. The two corresponding configurations are id=1 "{ref=IPTEL-UNC}" without modification, the other uses redundancy coding by PCMU as both primary and secondary encoding: id=2 "{codec=ref:G711-Red;secondary=PCMU, transport=ref:AVP}". Typically, transport address parameters such as the port number would also be provided but are omitted here for brevity.

During/after the negotiation phase, an actual configuration is chosen of out a number of alternative potential configurations, the actual configuration may refer to the potential configuration just by its "id", possibly allowing for some parameter modifications. Alternatively, the full actual configuration may be given.

If, from the above example, potential configuration #1 is chosen,, this could be expressed either in short form as "config=ref:1" or fully specified as id=1 "{ref=IPTEL-UNC}".

### 7.1.3 Constraints

Definitions specify media, transport, and other capabilities, configurations indicate which combinations of these could be used to provide the desired functionality in a certain setting.

There may, however, be further constraints within a system (such a CPU cycles, DSP available, dedicated hardware, etc.) that limit which of these configurations can be instantiated in parallel (and how many instances of these may exist). We deliberately do not couple this aspect of system resource limitations to the various application semantics as the constraints exist across application boundaries. Also, in many cases, expressing such constraints is simply not necessary (as many uses of the current SDP show), so additional baggage can be avoided where this is not needed.

Therefore, we introduce a "Constraints" section to contain these additional limitations. Constraints refer to potential configurations and to entity definitions and express and use simple logic to express mutual exclusion, limit the number of instantiations, and allow only certain combinations.

By default, the "Constraints" section is empty (or missing) which means that no further restrictions apply.

7.1.4 Session

The "Session" section is used to describe general parameters of the communication relationship to be invoked or modified. It contains most (if not all) of the general parameters of SDP (and thus will easily be usable with SAP for session announcements).

In addition to the session description parameters, the "Session" section also ties the various components to certain semantics. If, in current SDP, two audio streams were specified (possibly even using the same codecs), there was little way to differentiate between their uses (e.g. live audio from an event broadcast vs. the commentary from the TV studio).

This section also allows to tie together different media streams or provide a more elaborate description of alternatives (e.g. subtitles or not, which language, etc.).

Further uses are envisaged but need to be defined.

7.2 Syntax Proposal

In order to allow for the possibility to validate session descriptions and in order to allow for structured extensibility it is proposed to rely on a syntax framework that provides concepts as well as concrete procedures for document validation and extending the set of allows syntax elements.

SGML/XML technologies allow for the preparation of Document Type Definitions (DTDs) that can define the allowed content models for the elements of conforming documents. Documents can be formally validated against a given DTD to check their conformance and correctness. For XML, mechanisms have been defined that allow for structured extensibility of a model of allowed syntax: XML Namespace and XML Schema.

XML Schema allows to constrain the allowed document content, e.g.

for documents that contain structured data and also provide the possibility that document instances can be conformant to several XML Schema definitions at the same time, while allowing Schema validators to check the conformance of these documents.

Extensions of the session description language, say for allowing to express the parameters of a new media type, would require the creation of a corresponding XML schema definition that contains the specification of element types that can be used to describe configurations of components for the new media type. Session description documents have to reference the non-standard Schema module, thus enabling parsers and validators to identify the elements of the new extension module and to either ignore them (if they are not supported) or to consider them for processing the session/capability description.

It is important to note that the functionality of validating capability and session description documents is not necessarily required to generate or process them. For example, end-points would be configured to understand only those parts of description documents that are conforming to the baseline specification and simply ignore extensions they cannot support. The usage of XML and XML Schema is thus rather motivated by the need to allow for extensions being defined and added to the language in a structured way that does not preclude the possibility to have applications to identify and process the extensions elements they might support. The baseline specification of XML Schema definitions and profiles must be well-defined and targeted to the set of parameters that are relevant for the protocols and algorithms of the Internet Multimedia Conferencing Architecture, i.e. transport over RTP/UDP/IP, the audio video profile of RFC1890 etc.

The example below shows how the definition of codecs, transport-variants and configuration of components could be realized. Please note that this is not a complete example and that identifiers have been chosen arbitrarily.

```
<codec id="audio-basic"
             encoding="PCMU"
             sampling_rate="8000"
             channels="1"/>

<codec id="audio-L16-stereo"
             encoding="L16"
             sampling_rate="44100"
             channels="2"/>

<transport id="transport-RTP"
             transport="UDP"
             network="IPv4"
             framing="RTP"/>

<component id="c1">
  <config id="c1.1" ref="audio-basic transport-RTP">
    <override udp:port="6789"
                     receive-only="1"/>
    </config>
</component>
```

The example does also not include specifications of XML Schema definitions or references to such definitions. This will be provided in the next version of this draft.

A real-world capability description would likely be shorter than the presented example because the codec and transport definitions can be factored-out to profile definition documents that would only be referenced in capability description documents.

## 7.3 Mappings

A mapping needs to be defined in particular to SDP that allows to translate final session descriptions (i.e. the result of capability negotiation processes) to SDP documents. In principle, this can be done in a rather schematic fashion.

Furthermore, to accommodate SIP-H.323 gateways, a mapping from SDPng to H.245 needs to be specified at some point.

References

[1] Handley, M. and V. Jacobsen, "SDP: Session Description Protocol", RFC 2327, April 1998.

[2] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobsen, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[3] Schulzrinne, H., "RTP Profile for Audio and Video Conferences with Minimal Control", RFC 1890, January 1996.

[4] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A. and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, September 1997.

[5] Klyne, G., "A Syntax for Describing Media Feature Sets", RFC 2533, March 1999.

[6] Klyne, G., "Protocol-independent Content Negotiation Framework", RFC 2703, September 1999.

[7] Rosenberg, J. and H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction", RFC 2733, December 1999.

[8] Perkins, C. and O. Hodson, "Options for Repair of Streaming Media", RFC 2354, June 1998.

[9] Camarillo, G., Holler, J. and G. AP Eriksson, "SDP media alignment in SIP", Internet Draft draft-camarillo-sip-sdp-00.txt, June 2000.

[10] Rosenberg, J., Schulzrinne, H. and S. Donovan, "Establishing QoS and Security Preconditions for SDP Sessions", Internet Draft draft-ietf-mmusic-sdp-qos-00.txt, June 1999.

[11] Handley, M., Perkins, C. and E. Whelan, "Session Announcement Protocol", Internet Draft draft-ietf-mmusic-sap-v2-06.txt, March 2000.

[12] Kumar, R. and M. Mostafa, "Conventions for the use of the Session Description Protocol (SDP) for ATM Bearer Connections", Internet Draft draft-rajeshkumar-mmusic-sdp-atm-02.txt, July 2000.

[13] Quinn, B., "SDP Source-Filters", Internet Draft draft-ietf-mmusic-sdp-srcfilter-00.txt, May 2000.

[14] Beser, B., "Codec Capabilities Attribute for SDP", Internet

Draft draft-beser-mmusic-capabilities-00.txt, March 2000.

[15] Casner, S., "SDP Bandwidth Modifiers for RTCP Bandwidth", Internet Draft draft-ietf-avt-rtcp-bw-01.txt, March 2000.

## Authors' Addresses

Dirk Kutscher
TZI, Universitaet Bremen
Bibliothekstr. 1
Bremen 28359
Germany

Phone: +49.421.218-7595
Fax: +49.421.218-7000
EMail: dku@tzi.uni-bremen.de

Joerg Ott
TZI, Universitaet Bremen
Bibliothekstr. 1
Bremen 28359
Germany

Phone: +49.421.201-7028
Fax: +49.421.218-7000
EMail: jo@tzi.uni-bremen.de

Carsten Bormann
TZI, Universitaet Bremen
Bibliothekstr. 1
Bremen 28359
Germany

Phone: +49.421.218-7024
Fax: +49.421.218-7000
EMail: cabo@tzi.org

Acknowledgement

Funding for the RFC editor function is currently provided by the Internet Society.