US 20210273981A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2021/0273981 A1

Stach et al. (43) Pub. Date: Sep. 2, 2021

(54) METHOD FOR ESTABLISHING A COMMUNICATION CONNECTION WHICH IS SUITABLE FOR TRANSMITTING MEDIA STREAMS BETWEEN A FIRST RTC CLIENT AND A SECOND RTC CLIENT

(71) Applicant: **RingCentral, Inc.**, Belmont, CA (US)

(72) Inventors: **Thomas Stach**, Wien (AT); **Ernst Horvath**, Wien (AT); **Johannes Winter**, Gumpoldskirchen (AT)

(73) Assignee: **RingCentral, Inc.**, Belmont, CA (US)

(21) Appl. No.: **17/322,708**

(22) Filed: **May 17, 2021**

Related U.S. Application Data

(63) Continuation of application No. 15/318,726, filed on Dec. 14, 2016, filed as application No. PCT/EP2015/001116 on Jun. 2, 2015.

(30) **Foreign Application Priority Data**

Jun. 26, 2014 (DE) 102014009495.2

Publication Classification

(51) **Int. Cl.**
H04L 29/06 (2006.01)
H04L 29/08 (2006.01)

(52) **U.S. Cl.**
CPC ***H04L 65/1069*** (2013.01); ***H04L 65/1059*** (2013.01); ***H04L 67/42*** (2013.01); ***H04L 65/608*** (2013.01); ***H04L 67/02*** (2013.01); ***H04L 65/601*** (2013.01)

(57) **ABSTRACT**

The invention concerns a computer-implemented method for establishing a communication connection suitable for transmitting media streams from a first RTC client (**20**) to a second RTC client (**30**), comprising the following steps:

- the first RTC client (**20**) generates a request (**50**) to establish the communication connection, wherein the request (**50**) contains media-specific data and/or parameters for the first RTC client (**20**), and preferably also for the communication connection,
- the request (**50**) is adapted to the media-specific data and/or parameters of the second RTC client (**30**),
- the adapted request (**52**) is sent to the second RTC client (**30**),
- the second RTC client (**30**) generates a response (**60**) to the adapted request (**52**),
- the second RTC client (**30**) is configured using the adapted request (**52**) and the response (**60**), and the response (**60**) is also sent to the first RTC client (**20**), and
- the first RTC client (**20**) is configured using the adapted request (**52**) and the response (**60**).

The method is characterized in that

- the first RTC client (**20**) sends the request (**50**) to a web server (**40**) and the second RTC client (**30**) sends the response (**60**) to the web server (**40**), and
- the web server (**40**) adapts the request (**50**) and sends the adapted request (**52**) both to the second RTC client (**30**) and also back to the first RTC client (**20**), in addition to sending the response (**60**) both to the first RTC client (**20**) and also back to the second RTC client (**30**).

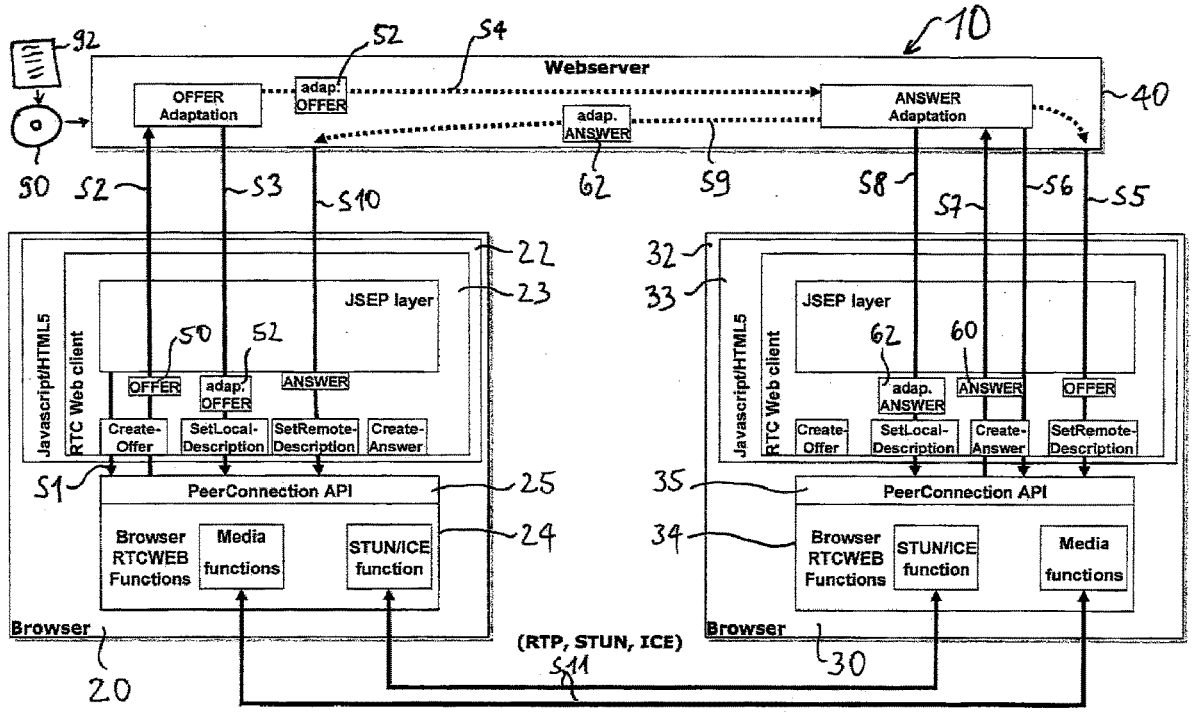


Fig. 1

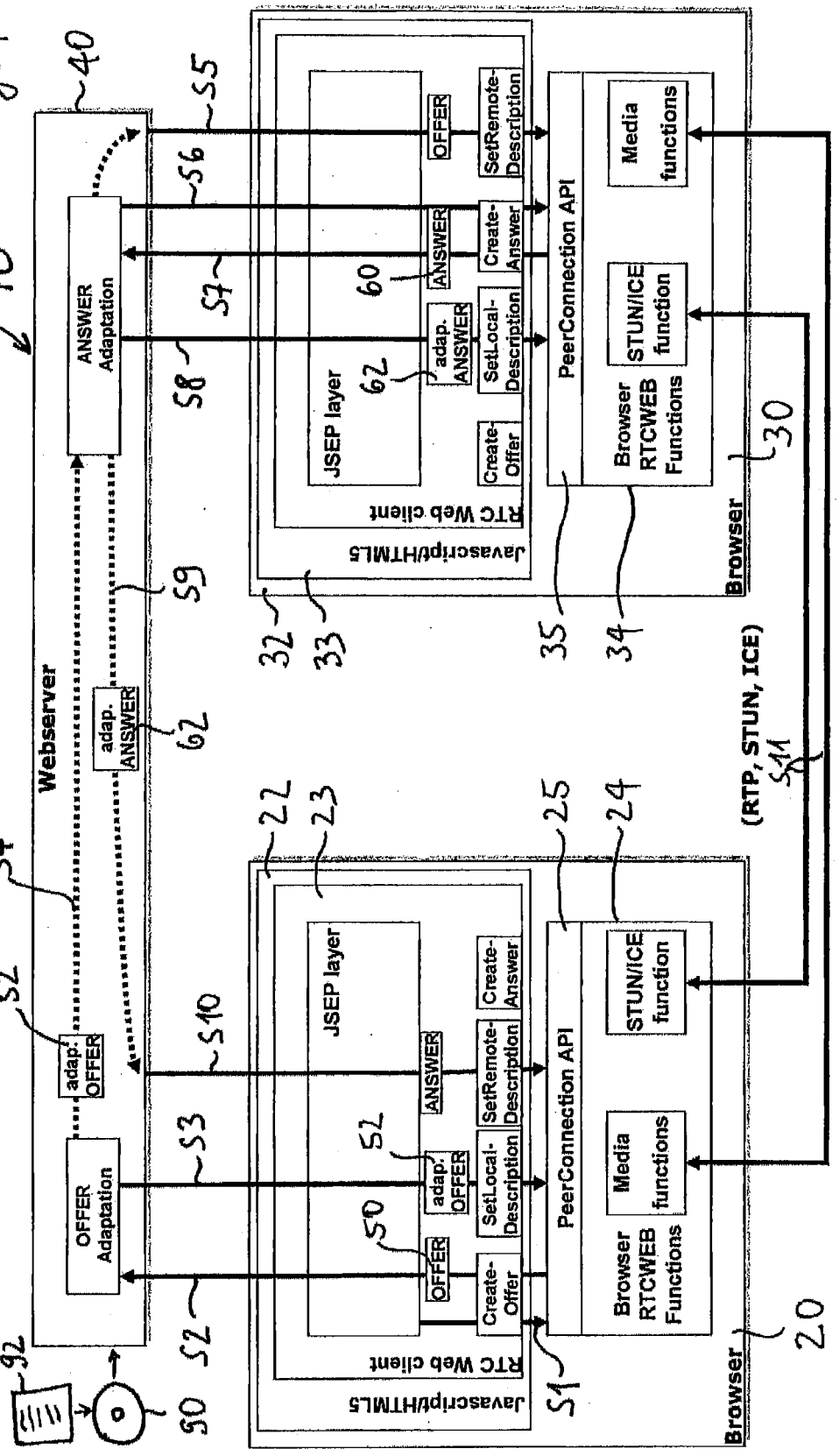

10
20
30
32
33
34
35
40
50
51
52
53
54
55
56
57
58
59
60
62
90
92
Webserver
Browser
Browser
JavaScript/HTML5
RTC Web client
JSEP layer
Create-Offer
SetLocal-Description
adap. ANSWER
Create-Answer
SetRemote-Description
OFFER
ANSWER
adap. OFFER
PeerConnection API
Browser RTCWEB Functions
STUN/ICE function
Media functions
OFFER Adaptation
ANSWER Adaptation
(RTP, STUN, ICE)
S1
S10
S11
S2
S3
S4
S5
S6
S7
S8
S9

Fig.2

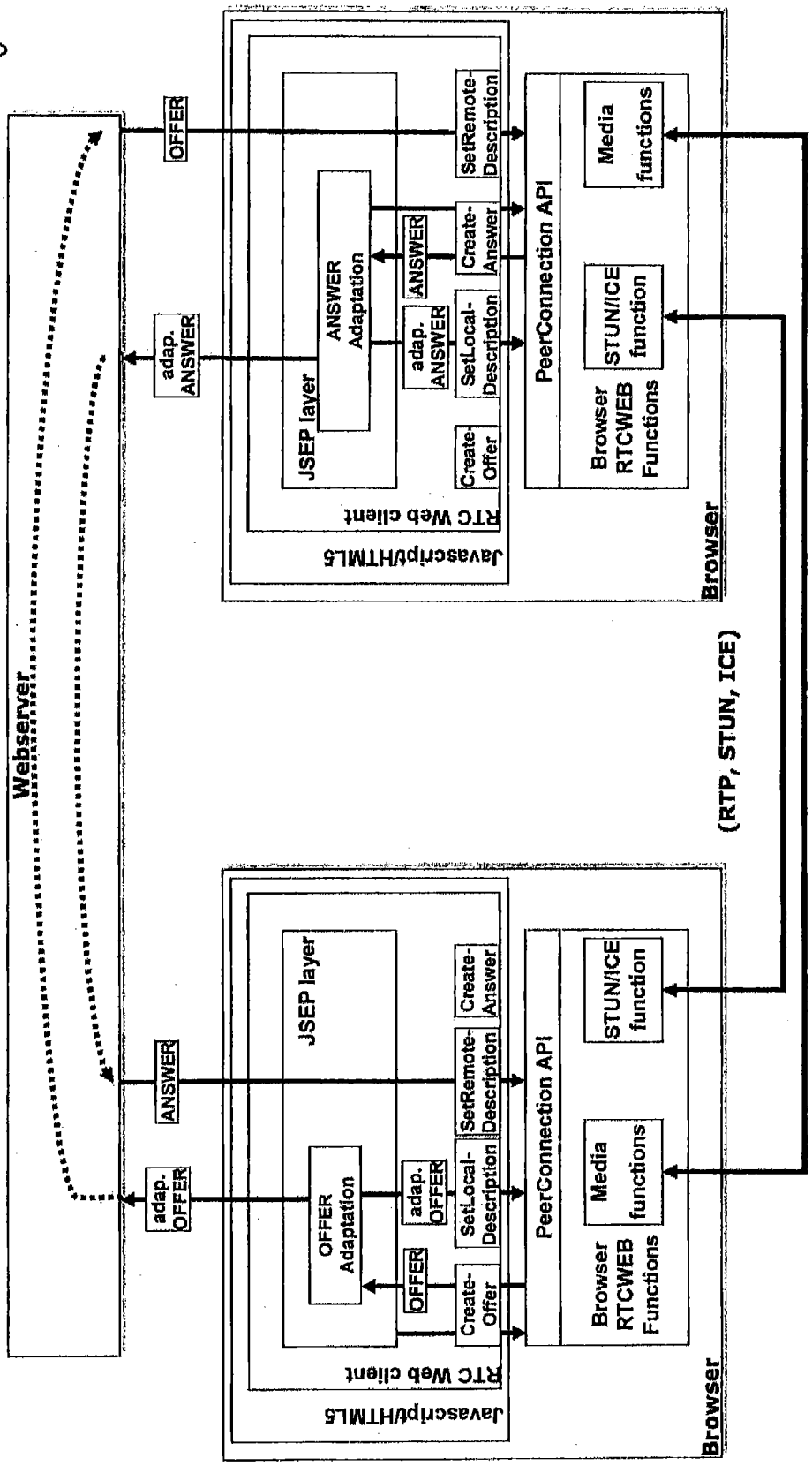

Webserver
OFFER
adap. ANSWER
ANSWER
adap. OFFER
JSEP layer
ANSWER Adaptation
OFFER Adaptation
SetRemote-Description
SetLocal-Description
Create-Answer
Create-Offer
adap. ANSWER
adap. OFFER
OFFER
RTC Web client
JavascripV/HTML5
PeerConnection API
Browser RTCWEB Functions
STUN/ICE function
Media functions
Browser
(RTP, STUN, ICE)

Fig. 3

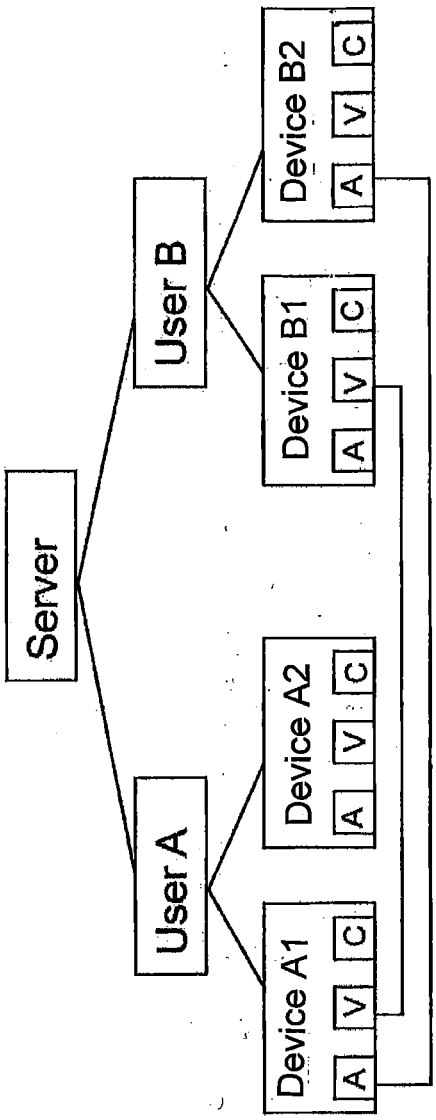
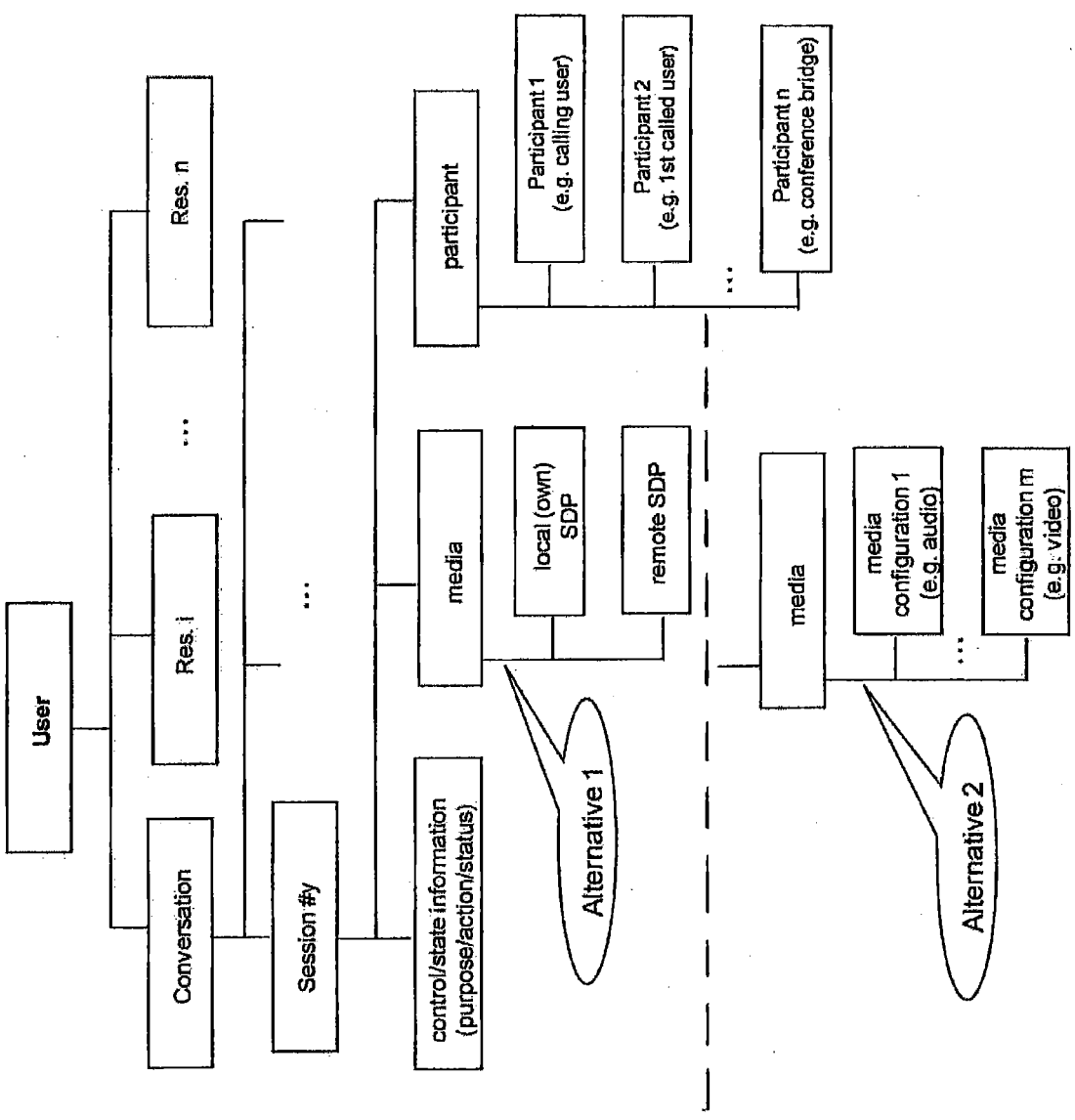

Server
User A
User B
Device A1
A
V
C
Device A2
A
V
C
Device B1
A
V
C
Device B2
A
V
C

Fig. 4


User
Conversation
Res. i
Res. n
Session #y
control/state information (purpose/action/status)
media
local (own) SDP
remote SDP
Alternative 1
participant
Participant 1 (e.g. calling user)
Participant 2 (e.g. 1st called user)
Participant n (e.g. conference bridge)
media
media configuration 1 (e.g. audio)
media configuration m (e.g. video)
Alternative 2

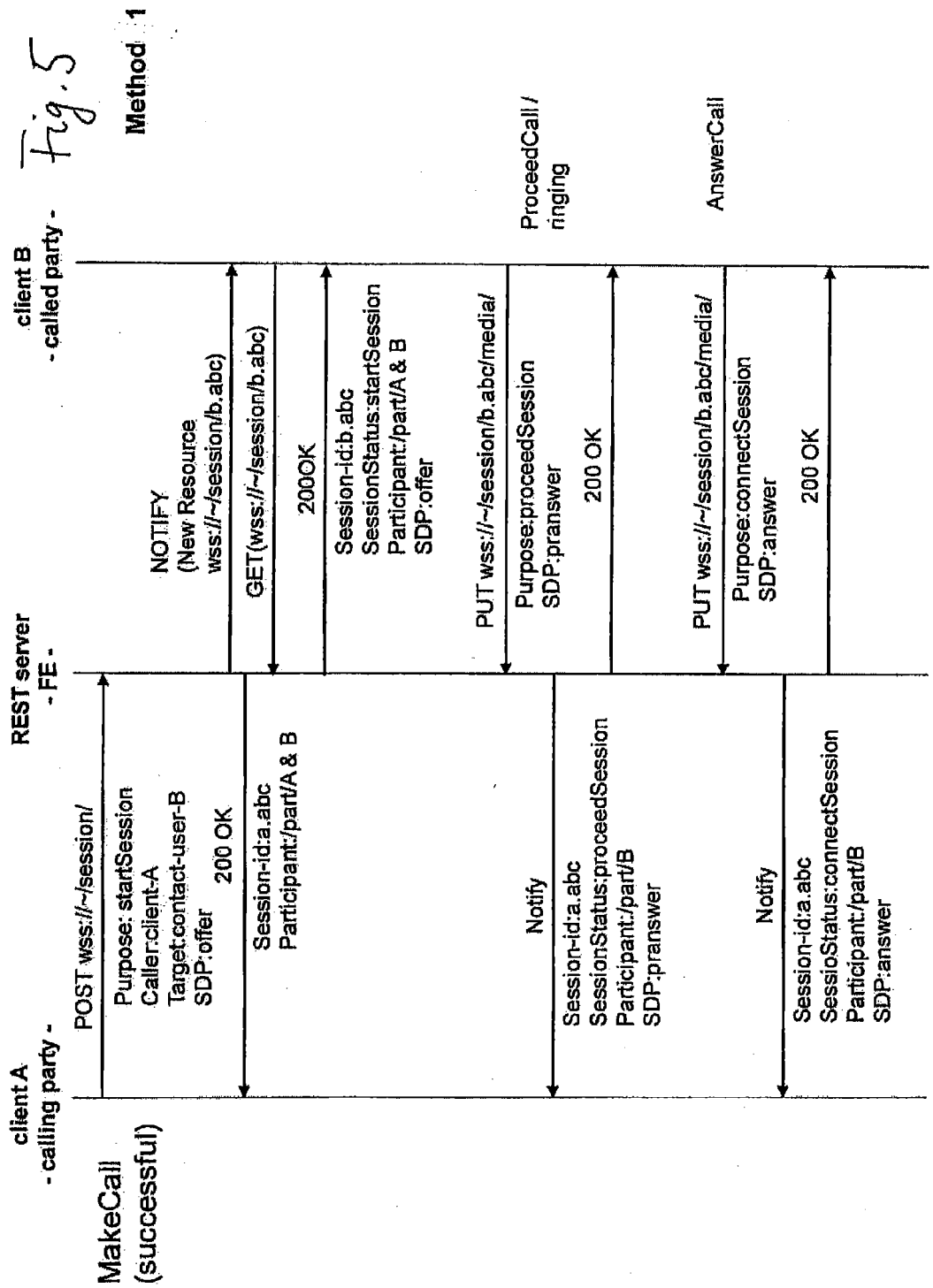

client A
- calling party -
REST server
- FE -
client B
- called party -
Method 1
MakeCall
(successful)
POST wss://~/session/
Purpose: startSession
Caller:client-A
Target:contact-user-B
SDP:offer
200 OK
Session-id:a.abc
Participant:/part/A & B
NOTIFY
(New Resource
wss://~/session/b.abc)
GET(wss://~/session/b.abc)
200OK
Session-id:b.abc
SessionStatus:startSession
Participant:/part/A & B
SDP:offer
PUT wss://~/session/b.abc/media/
Purpose:proceedSession
SDP:pranswer
ProceedCall /
ringing
200 OK
Notify
Session-id:a.abc
SessionStatus:proceedSession
Participant:/part/B
SDP:pranswer
PUT wss://~/session/b.abc/media/
Purpose:connectSession
SDP:answer
AnswerCall
200 OK
Notify
Session-id:a.abc
SessioStatus:connectSession
Participant:/part/B
SDP:answer

Fig. 5

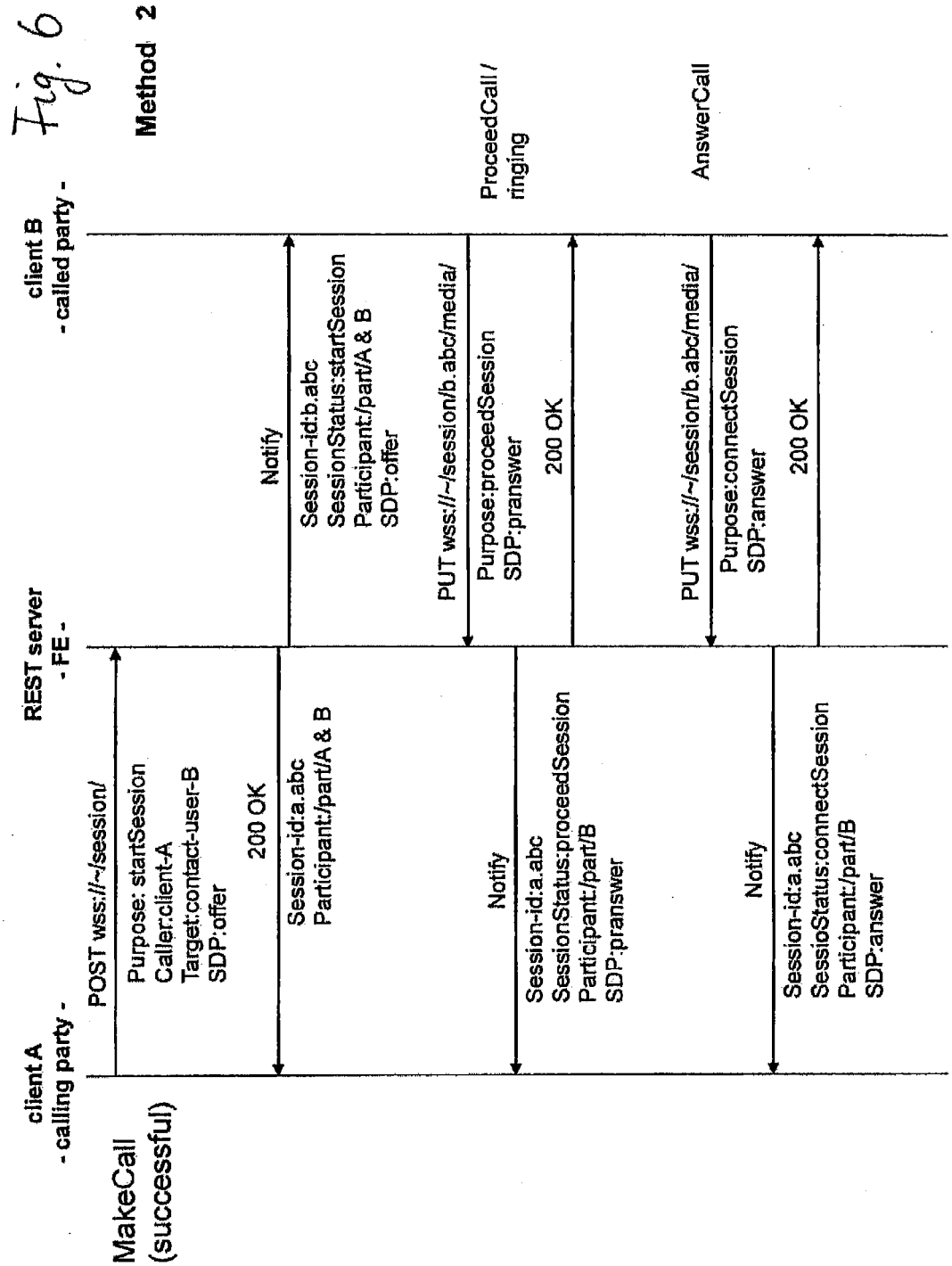


Fig. 6
Method 2
MakeCall
(successful)
client A
- calling party -
REST server
- FE -
client B
- called party -
POST wss://~/session/
Purpose: startSession
Caller:client-A
Target:contact-user-B
SDP:offer
200 OK
Session-id:a.abc
Participant:/part/A & B
Notify
Session-id:b.abc
SessionStatus:startSession
Participant:/part/A & B
SDP:offer
PUT wss://~/session/b.abc/media/
Purpose:proceedSession
SDP:pranswer
200 OK
ProceedCall / ringing
Notify
Session-id:a.abc
SessionStatus:proceedSession
Participant:/part/B
SDP:pranswer
PUT wss://~/session/b.abc/media/
Purpose:connectSession
SDP:answer
200 OK
AnswerCall
Notify
Session-id:a.abc
SessioStatus:connectSession
Participant:/part/B
SDP:answer

METHOD FOR ESTABLISHING A COMMUNICATION CONNECTION WHICH IS SUITABLE FOR TRANSMITTING MEDIA STREAMS BETWEEN A FIRST RTC CLIENT AND A SECOND RTC CLIENT

[0001] This invention concerns a method for establishing a communication connection suitable for transmitting media streams from a first Real-Time Communication (RTC) client to a second RTC client according to the preamble of claim **1**. This invention also concerns a computer program or computer program product implementing the method, a corresponding machine-readable data carrier with the computer program stored on it, and a telecommunication system according to the preamble of claim **10** for executing the method.

[0002] Based, among other things, on a WebRTC initiative (Real-Time Communication via the Worldwide Web) from the IETF (Internet Engineering Task Force) and W3C (World Wide Web Consortium), the JSEP (JavaScript Session Establishment Protocol) is often used for exchanging media-specific data and parameters between a web browser and a JavaScript-based application running on the browser, in order to establish an IP-based communication connection with another browser, an IP-based telephone, or a gateway in a telephone network. The JSEP is described in an IETF specification, and the current state of the work is documented in http://tools.iettorg/html/draft-ietf-rtcweb-jsep-02. The JSEP is based on the exchange of SDP-based media descriptions (SDP=Session Description Protocol, RFC [Request for Comments] 4566) and uses a two-stage signaling model. In the first stage, the browser's abilities (also called capabilities) are called up in the form of an SDP data set (using a CreateOffer command, for example). In a second stage, the browser is configured with the desired media parameters by providing it with a—possibly modified—SDP data set, which can be done using the SetLocalDescription command, for example.

[0003] The media parameters are then sent to the communication partner by a web server. After receiving the response with the media parameters from the communication partner, the web server places the media parameters into the JavaScript code in the browser, which is then configured with the partner's media parameters (using the SetRemoteDescription command, for example), to make communication possible.

[0004] The current JSEP specification http://tools.iettorg/html/draft-ietf-rtcweb-jsep-02#section-1 states that the previously described modification of the SDP data sets takes place in the JavaScript application. A procedure of this type is shown in FIG. **2** as an illustrative example, in which JSEP with JavaScript-based SDP modification is used.

[0005] However, the previously described procedure is sometimes not considered to be flexible enough.

[0006] This invention is based on the objective of further developing the previously described method so that it is more flexible and can be applied with greater versatility.

[0007] This objective is achieved with a computer-implemented method as in claim **1**, a computer program product configured to implement this method as in claim **8**, a machine-readable data carrier as in claim **9** with the computer program product stored on it, and a telecommunication system as in claim **10**. Additional advantageous embodiments of the invention are the subject matter of the dependent claims.

[0008] According to the invention, a first RTC client generates a request for a communication connection to be established. The request therefore contains media-specific data and/or parameters for the first RTC client, and preferably also for the communication connection. Then the request generated by the first RTC client is adapted to the media-specific data and/or parameters of a second RTC client, with which a communication connection is to be established. This adapted request is then transmitted to the second RTC client, and it in turn generates a response to the adapted request. The second RTC client is then configured using the adapted request and response. In addition, the response is also transmitted to the first RTC client, which in turn is configured using the adapted request and the response. The invented method is characterized in that the first RTC client sends the response to a web server—and specifically not "through a web server" to the communication partner—and in the same manner the second RTC client also sends the response to the web server. Also according to the invention, the web server—and not the communication partner's browser—processes the request and sends the adapted request both to the second RTC client and also back to the first RTC client, in addition to sending the response both to the first RTC client and also back to the second RTC client.

[0009] Obviously, according to the invention, there can also be multiple first RTC clients and/or multiple second RTC clients, which are sending and/or receiving different media (audio/video), for example.

[0010] This invention is therefore based on the concept of having the request (SDP modification in one example) changed not through the JavaScript code in a browser but rather in the web server. To do this, the appropriate data must be addressed to the corresponding web server, which is not used exclusively as the transit station for data and/or parameters, requests and responses.

[0011] In this case, it should be noted that the protocols between a browser and a web server do not necessarily have to be defined. However, the data and the codecs, etc. do have to be defined, which still does not require the use of the SDP. However, usually a browser sends an SDP to a WebRTC client, through a CREATE command, for example. For a better understanding of this invention, some abbreviations and explanations are summarized below:

[0012] (S)RTP=(Secure) Real-Time Transport Protocol, see RFC 3550, RFC 3711,

[0013] STUN=Session Traversal Utilities for NAT (Network Address Translation), see RFC 5389,

[0014] ICE=Interactive Connectivity Establishment, see RFC 5245,

[0015] REST=Representational State Transfer,

[0016] SIP=Session Initiation Protocol

[0017] a WebRTC client is typically a web browser,

[0018] a request is generated, for example, through an SDP-OFFER,

[0019] a response is generated, for example, through an SDP-ANSWER,

[0020] According to one advantageous version of the invented method, the web server adapts the response to the media-specific data and/or parameters of the first RTC client—as well as the communication connection if necessary—before the web server transmits the (adapted) response to the first RTC client and to the second RTC client. This is particularly advantageous when the situation has

changed between transmission and response, which can occur, for example, when the band width changes due to an additional video connection.

[0021] The invented method is particularly advantageous when multiple communication devices are assigned to one user, each having a first RTC client and/or a second RTC client, and when different media types are sent to different communication devices. This means that, for example, audio data can be sent to a smart phone with hands-free capability, but video data is sent to a tablet or notebook to be played on its larger screen. In such cases, the web server can modify the media parameters appropriately, based on knowledge of the various devices and the user's preferences. Therefore, this invention offers particular advantages when media streams include different types of media.

[0022] It can be advantageous when the connection between the first RTC client, the second RTC client, and the web server is a so-called Web-Socket connection or an HTTP-PUT request.

[0023] It can offer further advantages when so-called RESTful procedures are used in combination with asynchronous event reporting for the communication between the first RTC client, the second RTC client, and the web server. Here, for example, a call with multiple participants is modeled as a set of resources assigned to each of the participating users and managed centrally on the web server, wherein the web server then also coordinates these resources into a single overall picture of the call.

[0024] According to one embodiment of the invention, the Session Description Protocol (SDP) can be used to generate and transmit the request, the adapted request, the response, and, if applicable, the adapted response.

[0025] A computer program product configured to be suitable for executing the previously described method, as well as a machine-readable data carrier on which the computer program product is stored, are also considered part of this invention.

[0026] The problem on which the invention is based is further solved with a telecommunication system consisting of a web server, a first RTC client that is either connectable or already connected to the web server, and a second RTC client that is either connectable or already connected to the web server. The invented telecommunication system is therefore configured such that the web server, the first RTC client, and the second RTC client have equipment suitable for executing the previously described method. This equipment can consist, for example, of software components, processors suitable for running these software components, and/or hardware components or similar items.

[0027] Additional advantages, features, and characteristics of the present invention are presented in the following description of advantageous embodiments with reference to the drawing. The figures show schematically:

[0028] FIG. **1** a diagram of one embodiment of a telecommunication system according to the invention, in which the individual sequential steps can also be seen,

[0029] FIG. **2** a diagram of a telecommunication system according to the prior art and an overview representation of the sequential steps,

[0030] FIG. **3** a schematic diagram of a configuration in which audio and video data are transmitted from one participant to another participant,

[0031] FIG. **4** a diagram of a conference resource model, and

[0032] FIGS. **5** and **6** two examples of a call setup according to the invention.

[0033] According to FIG. **1**, the invented telecommunication system **10** consists of a web server **40**, a first RTC client **20**, and a second RTC client **30**, each of which is or at least can be connected to the web server **40**. Both RTC clients are or preferably include browsers or WebRTC browsers. Each of the RTC clients **20**, **30** is equipped with a first functional unit **22** or **32** and a second functional unit **24** or **34**. The first functional unit **22**, **32** includes JavaScript/HTML5, a JSEP layer, and an RTCWeb client **23** or **33**, and the second functional unit **24** or **34** includes the functions of a WebRTC browser (the RTCWeb reference in FIG. **1** is equivalent to the WebRTC designation) and contains a PeerConnection API **25** or **35**.

[0034] FIG. **1** also shows schematically an illustrated data carrier **90**, in the form of a CD, on which a computer program product **92** is stored and which can be read by the web server **40**, for example.

[0035] Below, with reference to FIG. **1**, an embodiment of the invented method for establishing a communication connection suitable for transmitting media streams from a first browser or RTC client **20** to a second browser or RTC client **30** is described. In other words, FIG. **1** refers to an invented method in which JSEP with web server-based SDP modification is used.

[0036] In a step S**1**, the WebRTC client **23** calls a PeerConnection API (API=Application Programming Interface) **25** of the second functional unit **24** in the browser **20** using the CreateOffer method.

[0037] In a step S**2**, as a result, an SDP data set is generated by the browser **20**, marked as a request **50** in the form of an SDP OFFER, and provided to the WebRTC client **23**. This request **50** contains media-specific data and parameters for the first browser **20** and the communication connection. The WebRTC client **23** forwards this request **50** to the web server **40**, for example through an already existing web-socket connection, an HTTP-PUT request, or other methods.

[0038] Next, in a step S**3**, the web server **40** can adapt or manipulate the SDP OFFER **50** in a variety of ways in a special "OFFER adaptation" function. Examples of this would be reducing the number of offered codecs or eliminating video streams, if there was not enough band width available, for example. This adapted response or SDP OFFER **52** is next sent back to the WebRTC client **23** in the browser **20** and forwarded through a SetLocalDescription method of the PeerConnection API **25** to the first browser **20**, which it then configures.

[0039] In a step S**4**, which can take place simultaneously with step S**3**, the adapted request **52** is sent to the second browser **30** in the form of an adapted SDP OFFER and is shared for the subsequent generation of an adapted response **62** in the form of a modified SDP ANSWER by the second browser **30** with a so-called "ANSWER adaptation" function.

[0040] In a step S**5**, the adapted request **52** is then forwarded to the second browser **30**, e.g., through an already existing web socket connection.

[0041] In a step S**6**, the adapted request **52** in the second browser **30** is forwarded by means of the SetRemoteDescription method of the PeerConnection API **35** to the browser **30**. To generate a response **60** in the form of an SDP ANSWER, the WebRTC client **33** calls up the Create-

Answer method of the PeerConnection API **35** and receives an SDP data set marked as SDP ANSWER.

[0042] In a step S**7**, the WebRTC client **33** forwards the response **60** back to the web server **40**, e.g., through an already existing web-socket connection, an HTTP-PUT request, or other methods.

[0043] Next, in a step S**8**, the SDP ANSWER **60** can be manipulated or adapted again in an "ANSWER adaptation" function with reference to the correspondingly adapted SDP OFFER **52**, and an adapted response **62** can be generated in the form of an adapted SDP ANSWER. This adapted SDP ANSWER **62** is then sent back to the WebRTC client **33** in the second browser **30** and forwarded through the SetLocalDescription method of the PeerConnection API **35** to the browser **30**, which it then configures. The described step S**8** is optional, since the response **60** or SDP ANSWER does not need to be adapted unless necessary. However, the step S**8** as such is necessary in order to call up the SetLocalDescription method.

[0044] In a step S**9**, which can occur simultaneously with step S**8**, the adapted SDP ANSWER **62** is sent back to the WebRTC client **23** in the first browser **20**.

[0045] In a step S**10**, the adapted SDP ANSWER **62** is forwarded by means of the SetRemoteDescription method of the PeerConnection API **25** to the browser **20**, which it then configures.

[0046] Both browsers **20**, **30** now know the media parameters of their respective counterparts and can exchange media streams, which is represented in a step S**11**. The abbreviations RTP, STUN, and ICE shown here stand for the exchange of the protocols used, as applicable.

[0047] FIG. **3** illustrates, as an example, a call in which a participant B receives audio data from a device B**2** and video data from another device B**1**, and the server combines the SDP data from these two end devices B**1** and B**2** into one SDP description before it is sent to participant A. According to this example, participant A can then receive both the audio data A and the video data V on the same device A**1**, i.e., from device B**2** or from device B**1** of participant B. As can be seen from this example, the device A**2** of participant A is not used at all. In this particular example, the data C represents data in general, such as that used for control signals, for example, or Instant Messaging, for example. The data C can therefore be any data except the aforementioned audio and video data. Obviously, many other examples of a connection with various media are possible.

[0048] FIG. **4** illustrates an exemplary model of a session resource used by a RESTful server. This exemplary embodiment is based on the concept of resources used in the Web and uses a method for communication between web servers and clients that involves the use of RESTful (Representational State Transfer) procedures with asynchronous event reporting instead of one of the known signaling protocols such as SIP (Session Initiation Protocol). REST describes a particular programming style for web applications. A description of its basic principles is available at http://de.wikipedia.org/wiki/Representational_State_Transfer, for example. REST is based on the abstract model of a resource on a web server, which can be unambiguously accessed using a URI (Uniform Resource Identifier). There is a set of standard commands for this purpose, designated as CRUD (Create, Read, Modify/Update, Delete), with which the resource can be generated, read, changed, and even removed. These commands are executed, for example, using the HTTP methods PUT, GET, POST, and DELETE. These RESTful CRUD operations are illustrated here, for example, with the following HTTP methods:

Create→HTTP POST

Read→HTTP GET

Update→HTTP PUT

Delete→HTTP DELETE

[0049] In this way the caller can place a new resource on the server, if he calls another participant and initiates a new communication session. This can be handled completely using the REST principles.

[0050] Beyond this point, the RESTful CRUD operations alone are not sufficient. Here, a new NOTIFY operation, previously unavailable in REST, is necessary, allowing the call recipient to also find out that he has been invited to the communication session or is being called right now. By means of the NOTIFY operation, the call recipient is informed that a new resource (i.e., a new communication session) was initiated.

[0051] Then the call recipient can also use the REST principles.

[0052] The NOTIFY operation is also necessary at the caller's end if the resource changes, because of the call recipient announcing his media parameters, for example. Additional details about this can be found at http://de.wikipedia.org/wiki/CRUD.

[0053] FIG. **4** illustrates how a call or an active communication relationship can be modeled as a session resource. It shows the user or participant as the primary resource and multiple sub-resources for one or more calls, media used (audio, video), other participants (e.g., in a conference call), etc. According to this example, a call with two to n participants is modeled as a set of resources assigned to each of the participating users and managed centrally on the web server. The web server coordinates these resources into a single overall picture of the call.

[0054] The sequences diagrammed above in FIG. **1** use the two containers positioned under "media" in FIG. **4** to exchange the adapted request (SDP OFFER) **52** and the (adapted) response (SDP ANSWER) **60** or **62** between the clients **20**, **30** and the web server **40** in the order shown in FIG. **1**. This is managed first by the standard commands Create, Read, Modify, Delete, included in the RESTful set, oriented on HTTP (Hypertext Transfer Protocol), and issued by the client to the server (synchronous request/response method), and second through asynchronous event reports from the server to the client (the aforementioned NOTIFY operation).

[0055] As previously illustrated, modified function distribution during SDP generation is possible according to the invention. It is also possible to implement resource-based control instead of using the customary signaling protocol.

[0056] In summary:

- **[0057]** The resources are located on a so-called back-end server,
- **[0058]** Each participant or user represents a root on the respective resource tree,
- **[0059]** One session resource #y is one instance of a participant's conversation resource (i.e., the respective conversation),

[0060] The sub-resource media describes the current media configuration per media type (alternative 2) or per end point (alternative 1),

[0061] The participant sub-resources are all participating users in a session,

[0062] A conference bridge is also modeled as a separate participant, e.g., with a media server that handles all media streams.

[0063] FIG. **5** illustrates a first method for initiating a call from a caller (Calling Party) through a REST server ((Functional Entity or FE) to a call recipient (Called Party). A GET request is sent to the resource, which was announced by NOTIFY. The related message 200 OK therefore contains all parameter values for the related instance of the resource and all sub-resources with their parameter values.

[0064] According to the method **2** illustrated in FIG. **6**, the NOTIFY message could even already contain the parameter values. This can eliminate the need for the GET request from the first alternative.

[0065] Obviously the NOTIFY command can include one or more resource URIs (with additional parameters if necessary). In the case of a resource URI without additional parameters, the client must then send one or more GET requests for each resource URI, in order to obtain all the information.

[0066] If the GET request contains a resource URI with additional parameters, then only changed parameters for each resource are received.

[0067] This applies similarly to the PUT request. If the PUT request contains only one resource URI without additional parameters, then a completely new resource is set up. For a resource URI with additional parameters, only the sub-resources are changed corresponding to the parameters.

[0068] By way of further explanation, the expression "SDP pranswer" in connection with a NOTIFY message refers to the concept of a "provisional answer" in JSEP, which is equivalent to a type of preview.

[0069] It should be noted that the features previously described as method steps (components of the invented method) can also similarly be considered as equipment features of the invented telecommunication system with no specific mention required. This correspondence of features also applies inversely for the elements of the telecommunication system described as equipment features.

[0070] It should further be noted that the features of the invention described by referencing the presented embodiments, for example web server, (Web)RTC clients, the type and configuration of the parameters used, control commands, protocols and hardware components, the arrangement of the individual components in relation to each other or the sequence of the respective process steps can also be present in other embodiments or variations hereof, unless stated otherwise or prohibited for technical reasons. Not all features of individual embodiments described in combination must necessarily always be implemented in any one particular embodiment.

LIST OF REFERENCE INDICATORS

[0071] **10**=Telecommunication system
[0072] **20**=First browser/RTC client
[0073] **22**=First functional unit
[0074] **23**=First WebRTC client
[0075] **24**=Second functional unit
[0076] **25**=PeerConnection API
[0077] **30**=Second browser/RTC client
[0078] **32**=First functional unit
[0079] **33**=Second WebRTC client
[0080] **34**=Second functional unit
[0081] **35**=PeerConnection API
[0082] **40**=Web server
[0083] **50**=Request/SDP OFFER
[0084] **52**=Adapted request/SDP OFFER
[0085] **60**=Response/ANSWER
[0086] **62**=Adapted response/ANSWER
[0087] **90**=Data carrier
[0088] **92**=Computer program product
[0089] S**1**-S**11**=Steps
[0090] A=Audio data
[0091] C=General data
[0092] V=Video data

1. Computer-implemented method for establishing a communication connection suitable for transmitting media streams from a first RTC client (**20**) to a second RTC client (**30**), comprising the following steps:

the first RTC client (**20**) generates a request (**50**) to establish the communication connection, wherein the request (**50**) contains media-specific data and/or parameters for the first RTC client (**20**), and preferably also for the communication connection,

the request (**50**) is adapted to the media-specific data and/or parameters of the second RTC client (**30**),

the adapted request (**52**) is sent to the second RTC client (**30**),

the second RTC client (**30**) generates a response (**60**) to the adapted request,

the second RTC client (**30**) is configured using the adapted request (**52**) and the response (**60**), and the response (**60**) is also sent to the first RTC client (**20**), and

the first RTC client (**20**) is configured using the adapted request (**52**) and the response (**60**),

characterized in that

the first RTC client (**20**) sends the request (**50**) to a web server (**40**) and the second RTC client (**30**) sends the response (**60**) to the web server (**40**), and

the web server (**40**) adapts the request (**50**) and sends the adapted request (**52**) both to the second RTC client (**30**) and also back to the first RTC client (**20**), in addition to sending the response (**60**) both to the first RTC client (**20**) and also back to the second RTC client (**30**).

2. Method as in claim **1**,

characterized in that the web server (**40**) adapts the response (**60**) to the media-specific data and/or parameters of the first RTC client (**20**), as well as the communication connection if necessary, before the web server (**40**) transmits the adapted response (**62**) to the first RTC client (**20**) and to the second RTC client (**30**).

3. Method as in claim **1** or **2**,

characterized in that multiple communication devices are assigned to one user, each having a first RTC client (**20**) and/or a second RTC client (**30**), and that different media types are sent to different communication devices.

4. Method as in claim **3**,

characterized in that the media streams contain different types of media.

5. Method as in one of the preceding claims, characterized in that a Web-Socket connection or an HTTP-PUT request is used for the connection between the first RTC client (**20**), the second RTC client (**30**), and the web server (**40**).

6. Method as in one of the preceding claims, characterized in that RESTful procedures in combination with asynchronous event reporting are used for the connection between the first RTC client (**20**), the second RTC client (**30**), and the web server (**40**).

7. Method as in one of the preceding claims, characterized in that the Session Description Protocol (SDP) is used to generate and transmit the request, the adapted request, the response, and the adapted response.

8. Computer program product (**92**) for executing the method according to one of the preceding claims.

9. Machine-readable data carrier (**90**) with a computer program product (**92**) according to claim **8** stored on it.

10. Telecommunication system (**10**) comprising

a web server (**40**),

a first RTC client (**20**) connected or connectable to the web server (**40**),

a second RTC client (**30**) connected or connectable to the web server (**40**),

characterized in that the web server (**40**), the first RTC client (**20**), and the second RTC client (**30**) have equipment for executing the method as in one of claims **1** to **8**, wherein the equipment consists of software components in particular.

* * * * *