

MEDIA ADAPTATION FOR LOST CONTENT RECOVERY IN REAL-TIME ONLINE MEETINGS

Inventor:

Hong Heather Yu

FIELD

[0001] The disclosure generally relates to improving the quality of audio/visual interaction in real-time meetings over communication networks.

BACKGROUND

[0002] The use of real-time video conferencing applications has expanded considerably in recent years. Typical video conferencing falls into two categories – group video conferences with two or more attendees who can all see and communicate with each other in real-time, and online presentations where one or more hosts use audio, visual and text to present information to a large group of attendees. Both categories rely on fast and reliable network connections for effective conferencing and presentations and can suffer quality when network bandwidth between the host and attendees fluctuates or is limited.

[0003] Generally, when network problems occur, attendees may lose the audio or video portions of the conference, or both. During a live conference, there is little participants or attendees, or the meeting application, can do to ensure that attendees do not miss portions of the conference which are interrupted by network issues.

[0004] Quality of experience (QoE) is a measure of a customer's experiences with a service and is one of the most commonly used service indicators to measure video delivery performance. QoE describes metrics that measure the performance of a service from the perspective of a user or viewer. Typically, video content is the most bandwidth intensive portion of a lecture-based presentation. Video quality will become even more bandwidth intensive when holographic, three dimensional or volumetric video conferencing services are used.

SUMMARY

[0005] One general aspect includes a computer implemented method of recovering real-time online meeting content which is lost. The computer implemented method includes: transmitting and receiving real-time meeting data including meeting content to and from participant devices in a data format during a real-time online meeting; and determining that real-time meeting data transmitted in a sequence between devices during the real-time online meeting has been lost. The method also includes receiving real-time recovery data in a different data format, the real-time recovery data containing real-time online meeting content replacing meeting content in the lost real-time meeting data. The method also includes rendering the real-time recovery data to replace real-time meeting content replacing the lost. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

[0006] Implementations may include any of the aforementioned implementations of the computer implemented method where the meeting content is in an audio/visual format and the different data format a non-audio/visual format. Implementations may include any of the aforementioned implementations of the computer implemented method where rendering may include rendering the real-time recovery data in at the same time as rendering real-time meeting data received after the lost real-time meeting data was lost. Implementations may include any of the aforementioned implementations of the computer implemented method where real-time recovery data may include a text transcription of the audio/visual data and the rendering may include rendering the text transcription of the audio/visual data the real-time recovery data in at the same time as rendering real-time meeting data received after the lost real-time meeting data was lost. Implementations may include any of the aforementioned implementations of the computer implemented method where meeting content is in an audio/visual format and the different data format may include a playback rendering

optimized format. The playback rendering optimized content may include audio/visual content having a sub-set of both the audio and visual data in the lost real-time meeting data. Implementations may include any of the aforementioned implementations of the computer implemented method where playback rendering optimized content may include audio/visual content having a sub-set of less than all of audio data in the lost real-time meeting data. Implementations may include any of the aforementioned implementations of the computer implemented method where playback rendering optimized content may include audio/visual content having null portions of the audio/visual content removed. Implementations may include any of the aforementioned implementations of the computer implemented method where real-time recovery data is received from a cache on a network device in a network environment. Implementations of the described techniques may include hardware, a method or process, or computer software on a computer-accessible medium.

[0007] Another aspect includes a computer implemented method of recovering real-time meeting content which is lost during an online presentation. The computer implemented method also includes transmitting and receiving real-time meeting data including real-time meeting content to and from participant devices in a data format during a real-time online meeting; generating real-time content recovery data in a different data format, the real-time recovery data replacing meeting content in the real-time meeting data. The method also includes forwarding the real-time recovery data to replace real-time meeting content which has determined to be lost in communication to a host device, the real-time recovery data including content replacing the lost content to at least one participant device.

[0008] Implementations may include any of the aforementioned implementations of the computer implemented method where the meeting content is in an audio/visual format and the different data format is a non-audio/visual format. Implementations may include any of the aforementioned implementations of the computer implemented method where meeting content is in an audio/visual format and the different data format may include a playback rendering optimized format. Implementations may include any of the aforementioned implementations of the computer implemented method where generating may include creating the playback rendering optimized

content with audio/visual content which is a sub-set of both the audio and visual data in the lost real-time meeting data. Implementations may include any of the aforementioned implementations of the computer implemented method where generating may include generating playback rendering optimized content with only audio data, and which a sub-set of less than all of the audio/visual content in the lost real-time meeting data. Implementations may include any of the aforementioned implementations of the computer implemented method where playback rendering optimized content may include audio/visual content having null portions of the audio/visual content removed. Implementations may include any of the aforementioned implementations of the computer implemented method where generating may include transforming the meeting content in the meeting data lost in communication to a host device into a transcription of the lost meeting data. Implementations may include any of the aforementioned implementations of the computer implemented method where generating may include generating a text transcription of the audio/visual data which is forwarded for rendering at the same time as rendering real-time meeting data received after the lost real-time meeting data was lost.

[0009] One general aspect includes a processing system in a network. The processing system includes a processor readable storage medium; a processor device including a first non-transitory memory storage may include instructions; and one or more processors in communication with the memory. The one or more processors execute the instructions to: route real-time meeting data transmitted between a source device and at least one participant processing device during a real-time online meeting, the real-time meeting data including meeting content to and from participant devices in a data format during the real-time online meeting; generate real-time recovery data in a different data format, the real-time recovery data replacing meeting content in the real-time meeting data; store the real-time recovery data; and forward the real-time recovery data to replace real-time meeting content which has determined to be lost in communication to at least one participant device, the real-time recovery data including content replacing the lost content to at least one participant device. Other embodiments of this aspect include corresponding computer systems,

apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

[0010] Implementations may include one or more aforementioned processing systems where the meeting content is in an audio/visual format and the different data format a non-audio/visual format. Implementations may include one or more aforementioned processing systems where one or more first processors execute the instructions to generate by transforming the meeting content in the meeting data lost in communication to a host device into a transcription of the lost meeting data. Implementations may include one or more aforementioned processing systems where real-time recovery data may include a text transcription of the audio/visual data which is forwarded for rendering at the same time as rendering real-time meeting data received after the lost real-time meeting data was lost. Implementations may include one or more aforementioned processing systems where meeting content is in an audio/visual format and the different data format may include a playback rendering optimized format. Implementations may include one or more aforementioned processing systems where playback rendering optimized content may include audio/visual content having a sub-set of both the audio and visual data in the lost real-time meeting data. Implementations may include one or more aforementioned processing systems where playback rendering optimized content may include audio/visual content having a sub-set of less than all of audio data in the lost real-time meeting data.

[0011] Another aspect includes a user equipment device. The user equipment device also includes a storage medium including computer instructions. The device includes one or more processors coupled to communicate with the storage medium, where the one or more processors execute the instructions to cause the system to: transmit and receive real-time meeting data including real-time live meeting content to and from participant devices in a data format during a real-time online meeting; determine that real-time meeting data transmitted between devices during the real-time online meeting has been lost; receive real-time recovery data in a different data format, the real-time recovery data replacing real-time live meeting content in the real-time meeting data which has been lost; and rendering the real-time recovery data to

replace the real-time meeting content in the lost real time meeting data. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

[0012] Implementations may include the user equipment device where the meeting content is in an audio/visual format and the different data format a non-audio/visual format. Implementations may include the user equipment device wherein the meeting content is in an audio/visual format and the different data format may include a playback rendering optimized format. Implementations may include the user equipment device wherein the playback rendering optimized content may include audio/visual content having a sub-set of both the audio and visual data in the lost real-time meeting data. Implementations may include the user equipment device wherein the playback rendering optimized content may include audio/visual content having null portions of the audio/visual content removed. Implementations may include the user equipment device wherein the playback rendering optimized content may include audio/visual content having a sub-set of less than all of the audio data in the lost real-time meeting data. Implementations may include the user equipment device wherein the rendering may include forwarding the real-time recovery data for rendering at the same time as rendering real-time meeting data received after the lost real-time meeting data was lost. The real-time recovery data may include a text transcription of the audio/visual data.

[0013] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter. The claimed subject matter is not limited to implementations that solve any or all disadvantages noted in the Background.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] Aspects of the present disclosure are illustrated by way of example and are not limited by the accompanying figures for which like references indicate the same or similar elements.

[0015] FIG. 1 illustrates an interface of an online conference application showing a first example of audio/visual information which may be presented in the interface.

[0016] FIG. 2 illustrates another interface of an online conference application showing a second example of audio/visual information which may be presented in the interface.

[0017] FIG. 3 illustrates an example of a network environment for implementing a real-time audio video conference or presentation.

[0018] FIG. 4 illustrates a general method in accordance with the technology for content recovery in real-time online meetings.

[0019] FIG. 5 is a ladder diagram illustrating data progression between a source or presenter participant's device and a meeting participant device in an embodiment.

[0020] FIG. 6 is a flowchart illustrating one embodiment selecting a recovery scheme and playback rendering in the method of FIG. 4.

[0021] FIG. 7 is a ladder diagram illustrating data progression between a source or presenter participant's device and a meeting participant device in an embodiment.

[0022] FIG. 8 illustrates one embodiment selecting a recovery scheme and playback rendering in the method of FIG. 4 using the packet delivery embodiment of FIG. 7.

[0023] FIG. 9 illustrates network environment showing locations of potential caches for recovery data in the network environment of FIG. 3.

[0024] FIG. 10 illustrates a traditional transmission order of ten segments/packets from a single queue where no recovery data is provided

[0025] FIG. 11 illustrates transmission order of ten segments/packets of real-time recovery data from a single queue.

[0026] FIG. 12 illustrates a transmission order for real-time recovery data when multiple sub-queues are used.

[0027] FIG. 13 is a flowchart of a method which performed at any of the network entities shown in FIGs. 3 or 9 which has a recovery data cache to determine whether to clear the recovery data cache.

[0028] FIG. 14 is a flowchart of a managed cache scheduling algorithm which may be performed by the management server or network nodes of FIGs. 3 and 9.

[0029] FIG. 15 illustrates the current state of source transmission and client receipt of real-time data for a real-time online meeting.

[0030] FIG. 16 is a timing diagram illustrating the timing of segments/packets transmitted between a source participant device and a receiving participant device.

[0031] FIG. 17 is a flowchart illustrating a process operating on a receiving device when data is received in the manner illustrated at FIG. 16.

[0032] FIG. 18 is a timing diagram illustrating the timing of segments/packets transmitted between a source participant device with client device queueing and playback where recovery data packets are transmitted in accordance with the embodiment of FIGs. 7 and 8.

[0033] FIG. 19 is a flowchart illustrating a process operating on a receiving device when data is received in the manner illustrated in FIG. 18.

[0034] FIG. 20 illustrates the relative timing for one example of how client device data queueing and playback correct for lost data in embodiments herein.

[0035] FIGs. 21 and 22 illustrate two alternative implementations of recovery process selection and processing.

[0036] FIG. 23 illustrates the presentation of transformed recovery data during a live meeting.

[0037] FIG. 24 is a ladder diagram illustrating data progression between a source or presenter participant's device and a meeting participant device, where multiple segments or packets are lost.

[0038] FIG. 25 is a flowchart illustrating a method for caching and entire real-time live meeting content stream.

[0039] FIG. 26 illustrates a managed cache scheduling algorithm which may be performed by the management server or network nodes of FIGs. 3 and 9.

[0040] FIG. 27 illustrates the timing of segments/packets transmitted between a source participant device and a receiver device, with receiver device playback of transformed data packets.

[0041] FIG. 28 illustrates the timing of segments/packets transmitted between a source participant device and a receiver device, along with the latencies introduced using the transformed recovery data method of FIG. 21.

[0042] FIG. 29 illustrates an example of a playback optimized segment/packet of data $W(n)$.

[0043] FIG. 30 illustrates the timing of segments/packets transmitted between a source participant device and a receiver device, with receiver device playback of playback optimized data segments/packets.

[0044] FIG. 31 illustrates the relative timing for one example of how client device data queueing and playback can correct for lost data in the present technology.

[0045] FIG. 32 is a flowchart of a general method performed at a participant device for implementing participant device-controlled data recovery.

[0046] FIG. 33 illustrates one embodiment of step 3260 of FIG. 32.

[0047] FIG. 34 illustrates another embodiment of step 3360 of FIG. 32.

[0048] FIG. 35 illustrates a method for participant side recovery with network device caching.

[0049] FIG. 36 illustrates a comparison of the relative timing of segments/packets transmitted between a source participant device and two receiving participant devices.

[0050] FIG. 37 illustrates the relative timing of segments/packets transmitted between a source participant device and two client devices Ca and Cb where the rendering method of FIG. 34 is used.

[0051] FIG. 38 is a general overview flowchart illustrating various embodiments of proactive initiation of real-time content recovery in an online meeting in combination with the herein-described real-time content recovery schemes.

[0052] FIG. 39 is a flowchart illustrating local and network caching which may be used with the proactive implemented content recovery.

[0053] FIG 40 illustrates two types of content rendering when a proactive, user-initiated break is initiated, based on the caching method illustrated in FIG. 39.

[0054] FIG. 41 illustrates client device content queuing and playback for a proactive break when transformed recovery data is utilized.

[0055] FIG. 42 illustrates client device content queuing and playback for a proactive break when playback optimized recovery data is utilized.

[0056] FIG. 43 illustrates an interface of an online conference application showing an example of simultaneously rendered live audio/visual information and recovery data.

[0057] FIG. 44 illustrates the relative timing of packets of live and recovery data when utilizing the interface and embodiment of FIG. 43.

[0058] FIG. 45 is a flowchart illustrating a method of content-aware proactive participant interruption.

[0059] FIG. 46 is a block diagram of a network processing device that can be used to implement various embodiments.

[0060] FIG. 47 is a block diagram of a network processing device that can be used to implement various embodiments of a meeting server or network node.

[0061] FIG. 48 is a block diagram illustrating an example of a network device, or node, such as those shown in the network of FIG. 3.

[0062] FIG. 49 illustrates one embodiment of a network packet implementation for enabling real-time data recovery.

DETAILED DESCRIPTION

[0063] The present disclosure and embodiments address performance improvements for real-time, online, audio-video conferencing and ensure a better QoE by recovering data which may be lost during a real-time online conference and providing the data to any participants of the conference for whom real-time conference data may be lost. Data may be lost due to communication interference, or proactively interrupted by a meeting participant on their device. The described embodiments may also be applied to broadcast presentations to multiple participants. Any reference herein to real-time conferences and conference data includes, as applicable, broadcast presentations.

[0064] In one aspect, systems and methods for compensating for lost real-time data in real-time online conferencing are presented. The communication interference may arise from network interference, low network bandwidth, low network throughput, excess network latency, network packet loss, network jitter, and/or other communication issues. Real-time content recovery comprises systems and methods for curing drops in the transmission and reception of media delivery, such as real-time online audio/video conferencing. Real-time content is that which is produced in actual time at the source, and which is delivered and consumed nearly simultaneously or in near actual time at a destination device (subject to transmission latency and delays). Real-time content comprises the information (audio, visual, textual) that is created by

participants in the meeting, and which is itself transmitted in data segments or data packets between devices in the meeting. In the real-time content recovery, recovery data delivery and recovery data playback work cooperatively to ensure real-time content which is lost (or where rendering is otherwise interrupted at a participant device) is rendered to a participant device in a seamless manner, thus ensuring a good quality of experience.

[0065] A first aspect includes transmitting recovery or “catch-up” content of a real-time, online, audio-video conference to one or more devices of participants of the real-time conference. Recovery content is transmitted in the form of recovery data replacing one or more segments, packets or packages of real-time audio/video conference data which have been either damaged or lost during the initial transmission (passive loss) or which have been missed by the receiver at the original playback time due to various reasons (including a proactive loss initiated by the receiver). Recovery data segments/packets replace segments/packets that were not received promptly for on-time rendering at the receiver or could not be rendered on time by the receiver for some other reason, hence causing interruption of continuous playback of the real-time content. In embodiments, the recovery content may be transmitted or retransmitted as recovery data in the original format or in a different format, such as transcoded or transformed format, or a recovery optimized format, to the receiver for rendering.

[0066] Recovery playback comprises the rendering of recovery content segments/packets which are received and rendered by the receiver at a time that is later than the originally expected render time for non-interrupted continuous rendering of real-time content. Recovery playback may be defined by two thresholds τ_1 and τ_2 , where τ_1 is the timeout threshold, comprising the latest reception time to avoid noticeable/perceptible errors in the real-time application and τ_2 is the latest time to receive the catch-up content per a defined user preference and/or system definition, such that the receiver will be able to catch-up the content in a reasonable or predefined catch-up period (a time frame T_{cp}), for an acceptable QoE during the real-time online conference.

[0067] Another aspect of real-time content recovery is content retransmission that is different from the existing transport or application layer packet retransmission. In traditional retransmission, in-time retransmission is used i.e., the retransmitted packet arrives before reaching the timeout threshold τ_n -- the arriving time t^* for a packet of sequence number $n=1$ is $t^* < \tau_1$ and is retransmitted in its original packet. In recovery transmission, recovery content is transmitted between (τ_1, τ_2) , i.e., conducted after timeout and the retransmission content format may be original, transformed or intelligently processed for optimized playback.

[0068] Another aspect is real-time meeting recovery using client-device controlled techniques. Caching occurs at a client participant device and/or in devices in the network environment, with these participant devices in embodiments recovering interrupted real-time data using techniques on the device.

[0069] Another aspect is real-time meeting recovery compensation when a meeting participant initiates a proactive pause or "break" in the meeting on their own device. Caching occurs at a client participant device and/or in devices in the network environment after initiation by the user, and meeting content can be recovered using various techniques.

[0070] Real-time meeting content as discussed herein may be divided into segments which may comprise one network packet or several packets. When reference is made to a packet or a segment, it should be understood that the principles discussed herein apply equally when referred to packets or segments. Standard network protocols govern the ordering of network packets, and time stamps and/or sequence numbers may be utilized to govern the ordering of segments when referring to transmission order, receipt order or rendering (playback) ordering. As used herein, ordering is described with respect to sequence numbers, but it should be understood that time stamps or any other segment/packet order tracking may be utilized in the described technology. All segments/packets transmitted between a source device and receiving participant devices in an ordered sequence may be referred to herein as a "stream" of data.

[0071] FIG. 1 illustrates an interface 100 of an online conference application showing a first example of audio/visual information which may be presented in the interface. Interface 100 is sometimes referred to as a “meeting room” interface, where a video, picture, or other representation of each of the participants is displayed. Interface 100 includes a presenter window 110 showing a focused attendee 120 who may be speaking or presenting and attendee display windows 130 showing other connected attendees of the real-time meeting. In this example, the presenter window 110 is showing an attendee but may also include text, video or shared screen information, as illustrated in FIG. 2. In addition, the attendee display windows may be arranged in a different location, as shown in FIG. 2, or may not be shown. In addition, the placement of the windows may differ in various embodiments. It should be understood that in embodiments, the presenter window may occupy the entire screen. The presenter window 110 may show a live action, real-time video of the presenter (the speaker), while other information is displayed on another portion of the display. In the application interface 200 shown in FIG. 2, audio/visual information 135 may be a motion video with accompanying text 145 provided in the presenter window (as shown) or different windows in the interface. It should be further understood that although eight attendees are illustrated in window 130 any number of users may be attending the conference or presentation.

[0072] FIG. 3 illustrates an example of a network environment for implementing a real-time conference application. Environment 300 includes a host processing device 310 which is associated with a meeting host. A meeting host generally comprises the meeting organizer who may subscribe to a service that provides real-time conferencing services using the online conference application. In real-time meetings the host is not necessarily always a meeting data source, and all participant devices may contribute to meeting data. In other embodiments, the host processing device may be a standalone meeting host connected via a network to participant processing devices. In addition, although host processing device 310 is illustrated as a notebook or laptop computer processing device, any type of processing device may fulfill the role of a host processing device. Also illustrated are three examples of participant devices, including a tablet processing device 312, a desktop computer processing

device 314 and a mobile processing device 316. It should be understood that there may be any number of processing devices operating as participant devices for attendees of the real-time meeting, with one participant device generally associated with one attendee (although multiple attendees may use a single device). Examples of processing devices are illustrated in FIGS. 43 – 45.

[0073] Also shown in FIG. 3 are a plurality of network nodes 320a - 320d and 330a – 330d and meeting servers 340a – 340d. The meeting servers 340a – 340d may be part of a cloud service 350, which in various embodiments may provide cloud computing services which are dedicated to the online conferencing application. Nodes 320a - 320d are referred to herein as “edge” nodes as such devices are generally one network hop from devices 310, 312, 314, 316. Each of the network nodes may comprise a switch, router, processing device, or other network-coupled processing device which may or may not include data storage capability, allowing cached meeting and recovery data to be stored in the node for distribution to devices utilizing the meeting application. In other embodiments, additional levels of network nodes other than those illustrated in FIG. 3 are utilized. In other embodiments, fewer network nodes are utilized and, in some embodiments, comprise basic network switches having no available caching memory. In still other embodiments, the meeting servers are not part of a cloud service but may comprise one or more meeting servers which are operated by a single enterprise, such that the network environment is owned and contained by a single entity (such as a corporation) where the host and attendees are all connected via the private network of the entity. Lines between the devices 310, 312, 314, 316, network nodes 320a - 320d, 330a – 330d and meeting servers 340a – 340d represent network connections which may be wired or wireless and which comprise one or more public and/or private networks. Examples of node devices are illustrated in FIGs. 43 and 44. As illustrated in FIG. 3, each of the participant devices 310, 312, 314, 316 may provide and receive real-time meeting data through one or more of the network nodes 320a- 320d, 330a – 330d and/or the cloud service 350.

[0074] FIG. 3 illustrates the flow of meeting data may be provided to client devices 312, 314, 316. As shown, each device may send and receive real-time meeting data. Each real-time meeting may include at least a live component, where the participant

speaks and/or presents information to others on the meeting, and may also include a pre-prepared, stored component such as a slide presentation or a shared screen or whiteboard application.

[0075] In one example, live meeting data 375 may be sent by a host or source device 310 through the host processing device's network interface and directed to the client computers though, for example, a cloud service 350, including one or more meeting servers 340a - 340d. In this example, the host device 310 is considered a meeting data "source" device. Within the cloud service 350 the data is distributed according to the workload of each of the meeting servers 340 and can be sent from the meeting servers directly to a client or through one or more of the network nodes 320a - 320d and 330a - 330d. In embodiments, the network nodes 320a - 320d and 330a - 330d may include processors and memory, allowing the nodes to cache data from the live meeting or presentation. In other embodiments, the network nodes do not have the ability to cache live meeting or recovery data. In further embodiments, meeting data may be exchanged directly between participant devices and not through network nodes or routed between participant devices through network nodes without passing through meeting servers. In other embodiments, peer-to-peer communication of real time content and recovery content may be utilized.

REAL-TIME MEETING CONTENT RECOVERY

[0076] FIG. 4 illustrates a general method in accordance with the technology for content recovery in real-time online meetings. At 410, one or more participants using a participant processing device is admitted to the real-time content recovery service. Admission control may be provided by the meeting service provider through use of a meeting application, and may be performed on a meeting-by-meeting or individual user basis. At 415, a prediction of expected jitter and/or packet loss in the network environment may optionally be made. At 415, the method may use this information to determine whether to cache recovery content or not, and if so whether to cache it at network nodes, the meeting servers, on local machines or on other cache enabled network devices. At 420, the meeting real-time content begins and participants in the meeting share meeting information through various means. The meeting is typically

begun by a meeting host (usually one of the participants) who begins the meeting using the meeting service 350.

[0077] Once the real-time online meeting has begun at 420, the method determines whether any segments/packets have not been delivered to any participants by checking for a packet or segment timeout. As discussed below, this can occur at individual devices, network nodes or meeting servers. Step 420 may also comprise determining if any segments/packets are corrupted or otherwise not suitable for rendering at a receiving device. While the embodiments herein may discuss “lost” segment/packets, the embodiments of real-time data recovery apply equally any received segments/packets which are unrenderable. At 425, if segments/packets are continually received within the timeout period, then the real-time meeting content is continually rendered on participant devices at 430 and, caching of segments/packets may occur at one or more of the network devices of FIG. 3. Whether and to what extent caching occurs generally depends on the system configuration and the real-time data recovery implementation embodiment. If the segment/packet timeout occurs at 425, a decision is made at 435 as to whether or not to initiate the real-time recovery procedure. The decision at 435 is based on predefined parameters including a system policy 465 which may be defined by the real-time meeting service provider, user preferences 470, and network status 475. If real-time recovery is not initiated at 435, then the method returns to step 430 and continues to render real-time meeting content with any dropped segments/packets not rendered. If real-time recovery is initiated at 435, then the system may optionally estimate the resources available to provide the recovery data for use in determining optimal data routing and recovery of data from caches throughout the network environment. As will be described below, there are various different forms of real-time recovery provided by the technology. Some of these methodologies require more resources than others, and the estimation at 440 considers the network bandwidth 480, computing resources available at 485, and cache availability 490.

[0078] At 445, the real-time content recovery scheme is selected. Again, multiple techniques for real-time meeting content recovery are described herein and one recovery method is selected at 445. At 450, if the content recovery scheme so

requires, content processing may occur. Various types of content processing are discussed below including no content processing, transformed recovery data and optimized recovery data. At 455, recovery data is sent to the participant processing device which as missing meeting content. At 460, a next segment/packet is incremented, and the method returns to step 425 to determine if additional segments/packets have been lost.

[0079] FIGs. 5 and 7 are ladder diagrams illustrating content data, recovery data and acknowledgements in one example of a data protocol used in embodiments of the technology. FIGs. 5 and 7 are described with respect to FIGs. 6 and 8, respectively, illustrating two different recovery content delivery methods. FIGs. 5 and 7 illustrate two embodiments of replacing segment/packet delivery with recovery content following loss. In a one embodiment illustrated in FIGs. 5 and 6, for a given lost segment/packet $X(n)$, all packets, except the lost packet/segment $X(n)$, are delivered as usual, i.e., in sync with the rest of the group, and the lost packet/segment $X(n)$ is transmitted as soon as bandwidth is available for a recovery data packet. In the embodiment of FIG. 7, the lost packet/segment $X(n)$ and packets/segments following $X(n)$ are cached in the cloud or edge, (designated herein as $X(m)$) and transmitted with a reception/lost delay latency $T_r(n)$. Transmission returns to normal the rendering device reaches a catch-up point τ_{cp} . In the embodiment of FIG. 6, delivery is in sequence, while in the embodiment of FIG. 5, delivery is not in sequence.

[0080] As used herein, a “catch-up” point is where playback/rendering of live content at a particular participant device begins to get back in sync with the rest of the participants of the live meeting (i.e. the original planned playback time had no loss or delay in playback occurred). The catch-up point is thus a point in time in the live meeting sequence when it suitable to skip, accelerate or omit live meeting recovery content, and return to rendering live meeting content, and may comprise (by way of example and without limitation): a point following a period of audio and/or visual live meeting content which is stagnant and/or silent (e.g. there are no participants speaking or presenting, and there is no or little change in video content); a point in time following an agreed upon, announced or pre-arranged “break” in the meeting during which participants may step away from their devices, or during which it is understood that

the avowed purpose of the meeting is suspended for a break period; a point in time designated by a participant at a participant device at which the participant indicates any non-rendered recovery data may be skipped or omitted; and/or a point in time at the end of the maximum acceptable delay T_{cp} .

[0081] FIG. 5 is a ladder diagram illustrating data progression between a source or presenter participant's device 502 and a meeting participant device 510, passing network nodes 504 and 508 and meeting servers 506. (In other embodiments, peer-to-peer communication of real time content and recovery content may be utilized.) FIG. 5 illustrates an example where one packet or segment is lost prior to reaching a participant device. In FIG. 5, time increases in the downward direction along the y-axis below each device. In FIG. 5, at 520, four segments/packets $X(1) - X(4)$ of real-time meeting data originate at source 502 and pass to edge 504, service host 506, edge 508 and participant device 510 where they are rendered. Service host 506 may acknowledge receipt of the segments/packets to the source 502 and device 510 may acknowledge receipt to the service host 506. Sequence 520 of packets $X(1) - X(4)$ represents a successful transmission and receipt of meeting data.

[0082] During time 525, which may be any time during the real-time meeting, four additional segments/packets $X(n-1)$, $X(n)$, $X(n+1)$, $X(n+2)$ of real-time meeting data originate at source 502 and passed to edge 504, service host 506, and edge 508. Packets $X(n-1)$, $X(n)$, $X(n+1)$, $X(n+2)$ are forwarded by host service 506 to edge 508, but at 530 segment $X(n)$ is lost due to any one of a number of network issues. As a result, device 510 recognizes that a segment/packet is missing in the sequence using standard techniques for recognizing dropped packets or service host designated techniques for recognizing lost segments of meeting data. In addition, no content acknowledgement for segment/packet $X(n)$ which is received at service host 506. At 535, device 510 initiates a service access request 555 which is transmitted through node 508 to service host 506 which acknowledges the service request and provides a recovery data replacement segment/packet $X(m)$ in the form of a replacement segment/packet which is then acknowledged by device 510. In this embodiment, $X(m)$ is a replacement segment/packet of the same for as the original real-time data segment/packet $X(n)$ it replaces. In one embodiment, replacement segment/packet

$X(n)$ is transmitted as soon as bandwidth at the device 510 is available for transmission. Additional real-time meeting data segments/packets are then forwarded after the replacement segment/packet $X(n)$ in the form of segment/packet $X(n+K+1)$, where “ K ” designates the number of segments following $X(n)$ the last received segment/packet at device 510. (In this example, $X(n+2)$ is the last segment/packet received at device 510 and K is equal to 1, but as demonstrated in FIG. 7, K can be any number.

[0083] In FIG. 5, recovery data is forwarded from a cache on the meeting service host 506, but in other embodiments, the data may be forwarded from a cache on the source device 502 or edge devices 504, 508, as well as the meeting service server devices.

[0084] FIG. 6 is a flowchart illustrating one embodiment of steps 445, 450, 455 and 460 of FIG. 4 using the packet delivery embodiment of FIG. 5 to recover real-time meeting data. In this embodiment, for a given lost segment/packet $X(n)$, all packets, except the lost packet/segment $X(n)$, are delivered as usual and the lost packet/segment $X(n)$ is transmitted as soon as there is available bandwidth for an extra data packet. At 645, the embodiment of real-time meeting data recovery selected is to transmit recovery data in the original segment/packet format when available, out of order, and render it when available at the receiving device in the correct transmission order with a delay. At 647, real-time meeting segments/packets are cached at one of the network devices, as described above. At 650, recovery data segments/packets are determined based on segments/packets which are not received at the participant device (510 in FIG. 5). This determination may be made from failure to receive and acknowledgement of one or more packets from the participant device and/or based on the service access request made by the participant device which did not receive the segments/packets. In other embodiments, edge device 508 or meeting service servers 506 may initiate recovery after the ACK for a segment/packet time out. After segments/packets are lost, lost segments/packets are forwarded from a data cache in one of the network elements at 655. At 655, the recovery data segment/packets are transmitted in their original format (in this embodiment) to the participant device that did not receive them as soon as available and out of order, for

rendering in original order as soon as packet is available. The method then proceeds to step 460 of FIG. 4.

[0085] FIG. 7 is a ladder diagram illustrating data progression between a source or presenter participant's device 502 and a meeting participant device 510, passing network nodes 504 and 508 and cloud host service 506. FIG. 7 illustrates an example where multiple segments/packets are lost prior to reaching a participant device. In FIG. 7, the lost packet/segment $X(n)$ and packets/segments following $X(n)$ are cached in the cloud or edge transmitted with a reception/lost delay latency $T_r(n)$. Transmission returns to normal once it reaches the catch-up point τ_{cp} , and thus there is a catch-up latency T_{cc} which is introduced in this embodiment. As in FIG. 5, in FIG. 7 time increases in the downward direction along the y-axis below each device. As in FIG. 5, in FIG. 7 at 520, four segments/packets $X(1) - X(4)$ of real-time meeting data complete a successful transmission and receipt of meeting data between the respective devices. During time 525, which may be any time during the real-time meeting, for additional segments/packets $X(n-1)$, $X(n)$, $X(n+1)$, $X(n+2)$ of real-time meeting data originating at source 502 are passed to edge 504, service host 506, and edge 508. Packets $X(n-1)$, $X(n)$, $X(n+1)$, $X(n+2)$ are forwarded by host service 506 to edge 508, but at 530 segments $X(n)$, $X(n+1)$, $X(n+2)$ are lost due to any one of a number of network issues.

[0086] At 650, a timeout τ_0 occurs and at 635, device 510 initiates a service access request which is transmitted through node 508 to meeting host server 506 which acknowledges the service request and provides replacement segments/packets $X(m)$, $X(m+1)$, $X(m+2)$ which are then then acknowledged by device 510. In one embodiment, replacement segment/packet $X(m)$, $X(m+1)$, $X(m+2)$ (which are cached versions of $X(n)$, $X(n+1)$, $X(n+2)$) are transmitted as soon as bandwidth at the device 510 is available for transmission. Additional real-time meeting data segments/packets are then forwarded after the replacement packets $X(m)$, $X(m+1)$, $X(m+2)$ in the form of segment/packet $X(n+K-1)$, $X(n+K)$, and $X(n+K+1)$.

[0087] It is noteworthy that in FIG. 7, because multiple segments/packets $X(n)$, $X(n+1)$, $X(n+2)$ are lost, the loss latency $T_r(n)$ will be slightly less than the catch-up latency $T_{cc}(n)$.

[0088] FIG. 8 illustrates one embodiment of steps 445, 450, 455 and 460 of FIG. 4 using the packet delivery embodiment of FIG. 7 where the lost packet/segment $X(n)$ and packets/segments following $X(n)$ are cached in the cloud or edge and transmitted with a reception/lost delay latency $T_r(n)$. In the embodiment of FIGs. 7 and 8, delivery of all packets (recovery data and new meeting data following the lost segments/packets) is in sequence, while in the embodiment of FIGs. 5 and 6, delivery is not necessarily in sequence.

[0089] In this embodiment, for a given lost segment/packet, or sequence of segments/packets (such as $X(n)$, $X(n+1)$, $X(n+2)$), all packets, except the lost packet/segments, are delivered as usual, and the lost packet/segment $X(n)$ is transmitted as soon as there is available bandwidth for an extra data segment/packet. At 845, the embodiment of real-time meeting data recovery selected is to transmit recovery data in the original segment/packet format when available, out of order, and render it when available at the receiving device in the correct transmission order with a delay. At 847, real-time meeting segments/packets are cached at one of the network devices, as described above. At 850, recovery data segments/packets are determined based on segments/packets which are not received at the participant device (510 in FIG. 7) and new packets are held so that the new and recovery data packets can be transmitted in sequence to the participant device. This determination may be made from failure to receive and acknowledgement of one or more packets from the participant device and/or based on the service access request made by the participant device which did not receive the segments/packets. After which segments/packets are lost, lost segments/packets are forwarded from a data cache in one of the network elements at 855. At 855, the recovery data segments/packets are transmitted in their original format in sequence with new data segments/packets to the participant device (510, for playback in original order. The method then proceeds to step 460 of FIG. 4

[0090] It should be understood that in the examples of FIGs. 5 and 6, service host 506 is performing certain steps of method 400 of FIG. 4 including, for example, steps 440 – 460, while participant device 510 performs steps 425 and 435. In other embodiments, any of these steps may be performed by the intermediate nodes, the source device, any participant device, or a specialized network element. In general, in

the network environments of FIGs. 3 and 9, the source device may perform the functions of device resource analysis, network resource analysis and content analysis and/or encoding, depending on the embodiment. Any edge device may perform network resource and analysis, content caching, and content transcoding and trans formatting in the embodiments described below. The service host device generally performs conference control and management, network resource analysis, data caching, content transcoding and trans-formatting, an intelligent content analysis and processing in the embodiments described herein. The client or participant device can perform device resource analysis network resource analysis and content decoding and monitoring.

[0091] In embodiments, service level protocols (not illustrated in FIGs. 5 and 7 or similar figures herein) may precede implementation of the real-time content recovery scheme. For example, each network device may communicate with the meeting host server to initiate both a connection request and acknowledgement and a service request and acknowledgement, before any real-time content or recovery data content is transmitted between the devices.

[0092] FIG. 9 illustrates the possible location of potential caches for recovery data in the network environment of FIG. 3. As illustrated above in FIGs 5 - 8, recovery data may be provided from the meeting server which caches recovery data in a recovery cache such as cache 802. It should be understood in FIG. 9 that each of the meeting servers 340a through 340d may include a recovery cache. In other embodiments none of the meeting servers 340a – 340d include recovery cache. In addition, each of the network nodes 320a through 320d and 330a through 330d may include recovery caches such as recovery caches 804 and 806. In other embodiments, none of the network nodes or only a portion of the network nodes include recovery caches. Client participant devices may also include recovery caches such as cache 808. It should be understood that each of the client participant devices may include a recovery cache or may not include any recovery cache in embodiments where caching is provided at the network nodes or the meeting servers. In embodiments, all participant devices may include a recovery cache to implement ordered playback of real-time data until a catch-up point τ_{cp} in order to remove the catch-up latency T_{cc} . In embodiments where

caching is provided at either the meeting servers or the network nodes, single or multiple queue caches may be utilized.

[0093] FIG. 10 illustrates a traditional transmission order of ten segments/packets 902 from a single queue where no recovery data is provided. In this figure, and in some of the figures herein where the sequence number is germane to the accompanying text, segments/packets are illustrated by encircled sequence numbers. In this embodiment, segments/packets 1 through 5 are provided in the queue and transmitted in order. Segments/packets are placed in a single queue based on their sequence number and delivery deadline. Five segments/packets 902 (ordered 1 – 5 in FIG. 10) are forwarded transmitted from a queue in order. When an acknowledgement of receipt for one of the five packets shown at 902 is not received and times out (indicating a dropped segment/packet as shown in FIGs. 5 and 7), it does not change the transmission order of the following packets. After the ACK timeout, any additional packets, in this case packets 6 – 10 are forwarded in order with no recovery data relative to a packet which dropped.

[0094] In FIG. 11, in accordance with the embodiment shown in FIG. 5, a recovery data segment/packet is transmitted as soon as bandwidth is available. In a single queue, as in FIG. 10, packets 1 through 5 are provided in the queue and transmitted in order before an expected ACK times out. After the ACK timeout, a replacement segment/packet for the dropped packet – in this example segment/packet number 4 – may be forwarded, but the transmission order changes and as shown at 904a, recovery data segment/packet 4 is inserted between packets 7 and 9 as shown at 907 and 908.

[0095] FIG. 12 illustrates a transmission order for recovery data when multiple sub-queues are used to implement the processes of FIGs. 5 - 8. Segment/packet are placed into two or more sub queues based on a segment/packet sequence number, type of the segment/packet, and delivery deadline. In one embodiment, recovery data segment/packet are placed in a separate sub queue different from the original data segment/packet sub queue, waiting for transmission. The transmission order and ACK timeout are the same as those illustrated in FIGs. 10 and 11, but in FIG. 12, recovery

data packet 1102 (priority 4) is placed in a separate sub queue. Priorities of the sub queues are defined based on application specific or system policy rules, as well as user preferences. The recovery data packet 1102, replacing packet sequence number 4, is forwarded whenever the outbound link becomes free for transmission. In one embodiment, when transmitting, the system first looks for the highest-priority sub queue, which may be the recovery data sub-queue, and transmits that data first.

[0096] When using multiple sub queues, one for original data and one for recovery data, network elements may utilize a scheduling algorithm to determine when transmission of recovery data should occur. A sample algorithm incorporates the transmission finishing time, packet segment size, output data rate and a priority factor. For example, given:

$F(i)$ = the transmission finishing time with $F(0)=0$ of the i^{th} packet/segment;

$F_o(i)$ and $F_{cc}(i)$ denote the transmission finishing time of the i^{th} packet/segment in the original content sub queue and the recovery data content sub queue respectively

$\beta(i)$ = the packet/segment size in bits of the i^{th} packet/segment

$\beta_o(i)$ and $\beta_{cc}(i)$ are the packet/segment size in bits of the i^{th} packet/segment in the original and the recovery data content sub queue respectively (where in some implementations, $\beta_o(i) = \beta_{cc}(i)$);

R = the output data rate of the current network node or the cloud; and

α_j = the priority factor of the j^{th} sub queue with $\alpha_j \in [1, J]$ and $\sum_1^J \alpha_j = 1$, where J is the total number of sub queues. Then the transmission finishing time is given by:

$$\begin{aligned} F(i) &= F(i-1) + \beta(i) * \alpha_j / R \\ &= \sum_{k=0}^{i-1} F(k) + \beta(i) * \alpha_j / R \\ &= \sum_{k=0}^i \beta(k) * \alpha_j / R \end{aligned}$$

[0097] Note that with different weighted queueing algorithms, α_j maybe defined differently and it maybe defined based on the application and system policy and rules and user preferences. When the system utilizes constant packet/segment size, then $\beta(i) = \beta$ for all packets/segments, thus

[0098]
$$F(i) = i * \beta * \alpha_j / R$$

[0099] With two sub queues, one for original content and one for recovery data content, with the priority factors of α_o and α_{cc} respectively, then

$$F_o(i) = \sum_{k=0}^i \beta_o(k) * \alpha_o / R \text{ and } F_{cc}(i) = \sum_{k=0}^i \beta_{cc}(k) * \alpha_{cc} / R$$

[00100] In the case of single active sub queue, $J=1$, then

$$F(i) = \sum_{k=0}^i \beta(k) * \alpha_1 / R$$

[00101] FIG. 13 illustrates a basic method which performed at any of the network entities shown in FIGs. 3 or 9 which has a recovery data cache to determine whether to clear the recovery data cache. In the method of FIG. 13, the entire conferencing content stream $X(n)$ is cached for $n \in [1, N]$ where N is the total number of segments/packets. In such an embodiment, when recovery is initiated, lost content segments/packets are fetched from the cloud/edge and delivered to the receiver device per system specification and user preference. This has the advantage of very simple cache management but may require additional storage space over more managed algorithms.

[00102] The method starts at 1205 with the segment/packet number “ n ” set equal to “1” at 1210. The method continuously checks for the end of the meeting at 1220 and if the meeting has not ended, at 1225 segments/packets are cached at the network element performing the method. The segment/packet numbers incremented at 1230 and the method continues caching additional segments/packets at 1225. When meeting has ended, a determination is made at 1240 as to whether or not it is time to clear the cache. Recovery data may be stored for a period of time after the meeting ends based on system policies 465 or user preferences 470. If it is time to clear the cache at 1250, the cache is cleared, and the method ends at 1260.

[00103] FIG. 14 illustrates a managed cache scheduling algorithm which may be performed by the management server or network nodes of FIGs. 3 and 9. The method of FIG. 14 takes advantage of the original content playback speed and the aforementioned “catch-up period” T_{cp} . T_{cp} comprises a predefined threshold that defines the maximum acceptable delay for recovery data content playback per the system policies and/or user preference. In one case, the recovery latency T_{cc} is less than or equal to the catch-up period ($T_{cc}(n) \leq T_{cp}$ for $n \in [1, N]$). The method of FIG. 14 may minimize the use of storage resources in network devices. The recovery latency is based in part on the rendering or playback time T_{ch} of cached segments/packets in a data cache. In the method of FIG. 14, if the playback time T_{ch} exceeds the predefined catch-up period T_{cp} , cached data is released.

[00104] At 1410, the segment/packet number “ n ” is set equal to 1 and the cached data segment/packet number, “ m ”, is set equal to 0. At 1415, determination is made as to whether or not the meeting has ended. If the meeting has not ended, then at 1420 an initial determination is made as to whether or not “ n ” is equal to 1. If so, then the cache delay T_{ch} is determined based on the sum of all segments/packets present in the current cache at 1425. At 1430, if T_{ch} is not greater than T_{cp} , then the current packet is cached by setting a cache sequence number $\check{n}$ (counting the order of caching) equal to the sequence number “ n ” added to the cache number “ m ”. At 1440, the method caches the packet $X(\check{n})$ and loops back to step 1445. At 1445, if n is equal to 1, then “ m ” is incremented at 1450. If “ n ” is not equal to one at 1445, then the method returns to step 1415.

[00105] At 1420, if n is equal to 1, or if at 1430 T_{ch} is greater than T_{cp} , the oldest packet in the cache $X(n)$ is released, and steps 1460 and 1465 (equivalent to 1435 and 1440) setting a cache sequence number $\check{n}$ equal to the sequence number “ n ” added to the cache number “ m ” and cache the packet $X(\check{n})$. At 1470, but n is incremented, and the method returns to 1445. This loop repeats, automatically flushing oldest segments/packets until the meeting ends at 1415. At that point, a determination is made at 1460 whether it is time to clear the cache at 1475 based on system policies 465 and user preferences 470. If so, the cache is cleared at 1480 and the method

ends at 1485. In other embodiments, joint cloud and edge caching algorithms can be used.

[00106] FIG. 15 illustrates the current state of source transmission and client receipt of real-time data for a real-time online meeting. In FIG. 15, time is illustrated on the X axis and increases from left to right. FIG. 15 illustrates a number of segments/packets 1502 along axis 1505 identified by a sequence number " n " (encircled) in a sequence $1 - n$ originating at a source participant device of a real-time meeting. If all segments/packets $1 - n$ are received at the receiver participant device, lines 1510 and 1520 would appear identical to line 1505, except that line 1510 would be delayed by any network delay introduced during transmission and line 1520 would be delayed by the network delay and any buffer latency at the receiving device.

[00107] However, in FIG. 15, four packets sequence numbered 4 – 7 are not received at a participant receiver device and as a result a playback gap of $T_r(4)$ is experienced by the participant. Once the gap has passed, there is no delay in subsequent playback. However, participant users experience the playback gap and may miss important meeting information. In general, the time needed to transmit a packet is a function of the network delay, which is generally equal to the propagation delay plus transmission delay, which are themselves a function of link distance, link speed, packet size and bandwidth.

[00108] FIG. 16 illustrates the timing of segments/packets transmitted between a source participant device and a receiving participant device without loss, a receiver participant device with loss and recovery data, with device queueing, and receiving participant device rendering (or playback) where recovery data segments/packets are transmitted in accordance with the embodiment of FIGs. 5 and 6 (recovery segments/packets arriving out of sequence). Line 1610 shows data segments/packets $X(1) - X(n)$ transmitted from a source device over time. Line 1620 shows arrival at a first participant device, for example device 312 in FIG. 3, without data loss and at a slightly later time relative to transmission due to network delay. Line 1630 illustrates segments/packets arriving at a second participant device with segment/packet $X(4)$ being lost in transmission. In accordance with the embodiment of FIGs. 5 and 6,

recovery data content 1635 in the form of segment/packet $X(4)$ is received at a time between segments/packets $X(5)$ and $X(6)$. The receiver participant device rendering/playback illustrated in line 1640 is configured to wait until segment/packet $X(4)$ is received before initiating playback, maintaining the sequence of transmitted segments/packets but introducing a playback rendering delay. At 1650, meeting rendering returns to normal (in sync with other participants) once the meeting reaches a catch-up point. In one embodiment, the receiver playback at 1640 may implement playback speed or simply skips ahead to render real-time data as received when detecting a break or lull in information in the data stream (as further described with respect to additional embodiments below).

[00109] FIG. 17 is a flowchart illustrating a process operating on a receiving device when data is received in the manner illustrated at 1630. At 1710, as illustrated in FIG. 5, a participant device (device 510 in FIG. 5) detects a segment/packet timeout and initiates a service access request. As noted above, in other embodiments, a participant device need not send a service access request, but the service may be initiated automatically by a network device. At 1720, the receiving participant device may pause meeting rendering and cache any new packets received before recovery data is received. (In FIG. 16, this would comprise packets $X(5)$ - $X(10)$.) At 1730, recovery data in the same format as live meeting data is received and at 1740, the participant device renders meeting playback of the segments/packets in sequence as soon as possible. At 1750, as soon as a catch-up point is detected, the participant device returns to rendering live meeting data in sync with other participant devices in the real-time online meeting. A catch-up point may be detected by a pause in audio, a defined meeting break by the host, or other means.

[00110] FIG. 18 illustrates the timing of segments/packets transmitted between a source participant device with client device queueing and playback where recovery data packets are transmitted in accordance with the embodiment of FIGs. 7 and 8 (recovery segment/packet and new segments/packets arrive in sequence). Line 1810 shows data packets $X(1)$ - $X(n)$ transmitted from a source device over time. Line 1820 shows arrival at a first participant device, for example device 312 in FIG. 3, without packet loss and at a slightly later time relative to transmission due to network delay.

Line 1830 illustrates packets arriving at a second participant device with packet $X(4)$ being lost in transmission. In accordance with the embodiment of FIGs. 7 and 8, other network elements ensure delivery of recovery data content in the form of segment/packet $X(4)$ in sequence, before packet $X(5)$. Thus, the receiver participant device playback at 1840 is the same as illustrated in line 1830 with a break in the sequence of transmitted segments/packets but introducing a playback delay. At 1850, playback returns back to normal once the meeting reaches a catch-up point. In one embodiment, the receiver playback at 1830 increases playback speed or simply skips ahead to render real-time data as received when detecting a break or lull in information in the data stream (as further described below).

[00111] FIG. 19 is a flowchart illustrating a process operating on a receiving device when data is received in the manner illustrated at 1840. At 1910, as illustrated in FIG. 7, a participant device (device 510 in FIG. 5) detects a segment/packet timeout and initiates a service access request. At 1920, the receiving participant device may pause meeting rendering and wait for recovery data and new real-time segments/packets. At 1930, recovery data in the same format as live meeting data, as well as new real-time meeting data comprising packets in sequence following the recovery segment/packets, are received and at 1940, playback of the meeting occurs using the sequence/packets in received order. At 1950, as soon as a catch-up point is detected, the participant device returns to rendering live meeting data.

[00112] FIG. 20 illustrates the relative timing for one example of how client device data queueing and playback can correct for lost data in the present technology. In FIG. 20, a single packet/segment is lost; however, it will be understood that the principles illustrated in FIG. 20 are similar for multiple lost segments/packets. Line 2010 illustrates a series of segments/packets 2020 identified by a sequence number " n " (encircled). As illustrated therein, the sequence of segments/packets 2020 may be received out of sequence. In this example, segment/packet sequence number 5 is delivered before sequence number 3 and segment/packet sequence number 10 before sequence number 9. Buffer latency is introduced on the playback device which ensures that even received packets are rendered in a correct order. When a time out period $T_w(4)$ indicates that packet sequence number for has not been received,

recovery is initiated, and recovery packet sequence number 4 is received after sequence 9. A recovery buffer latency T_{cb} is introduced so that packet sequence numbers 4 – 10 can be rendered in sequence until a catch-up point. The recovery playback is delayed by a playback gap 2025.

MEDIA ADAPTATION LOST CONTENT RECOVERY IN REAL-TIME ONLINE MEETINGS

[00113] A further aspect of this technology includes processing of the recovery data in a number of forms. FIGs. 21 and 22 illustrate two alternative implementations of recovery process selection (step 450) and processing (step 455). In the methods of FIGs. 21 and 22, recovery data is provided in a different data format than the originally transmitted live-meeting content data format. As described below, these different data formats comprise a transformed (or trans-formatted) data format and a rendering-optimized (or “optimized) data format. In embodiments, other types of different recovery data formats may be used.

[00114] Generally, in the method of FIG. 21, rather than providing recovery content data in its original format, the data is transformed into another format – referred to herein as trans-formatting – to produce transformed recovery data. Examples of trans-formatting include converting audio data to text data, converting video data to a series of lower resolution images, converting audio/visual data to text-only data, and the like. The transformed recovery data comprises data having a smaller size than that of the original data (or original format recovery data), and in some cases such recovery data can be rendered simultaneously with real-time data from the meeting.

[00115] Thus, in FIG. 21, at 2145, the selected method of recovery for dropped real-time data segments/packets comprises recovery of content by transformed recovery data. At 2147, an initial determination is made as to whether transformed recovery content is available. In embodiments, trans-formatting may occur on all source transmitted data at one or more of the network devices in the network embodiment. If the transformed recovery data is not available, the transformation is performed at 2155. Transformed recovery data designated $W(n)$ is then forwarded to the device where data segments/packets were lost.

[00116] In the method of FIG. 21, the transformed recovery data delivery (step 2150) resembles the delivery described above with respect to FIG. 5 except that the “recovery forwarding” of data will comprise transformed recovery data.

[00117] FIG. 23 illustrates one example of the rendering of transformed recovery data in a user interface during a live meeting. In this example, the lost or corrupted real-time meeting data has been transformed into text, and is overlaid on the current presenter 120 who may or may not be the person who generated the audio which has been transcribed. In other embodiments, the transformed recovery data can be presented in a separate window from the presenter.

[00118] Another form of processing of recovery data comprises intelligent processing of recovery data to remove elements of audiovisual data from the recovery data that are not needed for a complete understanding of the data in real-time. In the method of FIG. 22, playback optimized recovery data is created by intelligently processing the dropped segments/packets to remove, compress or otherwise optimize the data to speed up playback rendering of the data without loss of information to the participant. For example, intelligent processing comprising removing pauses, silent periods, repeated information, and non-crucial content to effectively speed up playback of both dropped segments/packets for which recovery content is generated and, in one implementation, segments/packets which follow the recovery content in order to provide a more rapid return to real-time data rendering.

[00119] Thus, in FIG. 22, at 2245, the selected method of recovery for dropped real-time data segments/packets comprises recovery of content by playback optimized recovery data. At 2247, an initial determination is made as to whether playback optimized recovery content (and in embodiments, intelligently processed real-time data) is available. In embodiments, intelligent processing may occur continually on all source transmitted data at one or more of the network devices in the network embodiment and playback optimized data segments/packets $X'(n)$ remain cached for use in recovering real-time data lost during transmission. If the playback optimized data is available, it is forwarded at 2250 to the device where data segments/packets were lost. If the playback optimized recovery data is not available, the intelligent

processing is performed at 2155. A determination is then made at 2156 as to whether real-time data following the missing data needs to be processed as well. This can be determined based on system policies and user preferences. If so, processing of real-time data following the missing data occurs at 2257 and is forwarded along with the intelligently processed recovery data at 2258.

[00120] FIG. 24 is a ladder diagram illustrating data progression between a source or presenter participant's device 502 and a meeting participant device 510, passing network nodes 504 and 508 and cloud host service 506 using the method of FIG. 22. FIG. 24 illustrates an example where multiple segments/packets are lost prior to reaching a participant device. In FIG. 24, time increases in the downward direction along the y-axis below each device.

[00121] During time 525, which may be any time during the real-time meeting, four segments/packets $X(n-1)$, $X(n)$, $X(n+1)$, $X(n+2)$ of real-time meeting data originate at source 502 and pass to edge 504, service host 506, and edge 508. Packets $X(n-1)$, $X(n)$, $X(n+1)$, $X(n+2)$ are forwarded by host service 506 to edge 508, but at 2430 segments/packets $X(n)$, $X(n+1)$, $X(n+2)$ are lost due to any one of a number of network issues. As a result, device 510 recognizes that a segment/packet is lost and at 2435, device 510 initiates a service access request, (or one is automatically generated by another network device after an ACK receipt timeout). The service access request is transmitted to node 508 and meeting service server 506, which acknowledges the service request and provides an playback optimized recovery data packets $X'(n)$, $X'(n+1)$, $X'(n+2)$ which contain processed recovery content which maximizes the information from the real-time meeting in as compressed a form as possible. Playback optimized processed packets $X'(n+k)$, $X'(n+k+1)$, continue until a catch-up point is reached. In FIG. 24, initial intelligent recovery data is $X'(n)$, $X'(n+1)$, $X'(n+2)$ forwarded from a cache on the meeting service host 506, but $X'(n+k+1)$, $X'(n+k+2)$,... begin at the source as $X(n+k)$, $X(n+k+1)$ as real-time packets which are converted by the meeting servers 506. As such, even those packets generated after data is lost can be processed as optimized recovery data until the catch-up point is reached. As noted herein, one or more of the network devices in the network

environments discussed above with respect to FIG. 3 and FIG. 9 may include a cache to store real-time data segments/packets.

[00122] FIG. 25 is a flowchart illustrating a method for caching an entire conferencing content stream of $X(n)$ segments/packets for $n \in [1, N]$ where N is the total number of packets/segments in a live meeting data stream. In such an embodiment, when recovery is initiated, intelligently process segments/packets, or transformed content segments/packets are fetched from the cloud/edge and delivered to the receiver device per system policies 465 and user preferences 470. In FIG. 25, steps 2510, 2514 and 2518 are equivalent to steps 1210, 1220, and 1225 of FIG. 14. Once $X(n)$ is cached, trans-formatting processing may occur at 2522 to generate transformed recovery data in the form of transformed segment/packet $W(n)$ which is cached at 2526. In addition, or in the alternative, intelligent processing to produce an optimized replacement segment/packet $X'(n)$ is performed at 2530 to generate optimized recovery data and the segment/packet $X'(n)$ is cached at 2534. The segment/packet number " n " is incremented at 2541 and the method loops to step 2514 to determine if conferencing has ended. When conferencing ends at 2514, steps 2546, 2550 and 2554, equivalent to steps 1240, 1250 and 1260 of FIG. 12, are performed.

[00123] FIG. 26 illustrates a managed cache scheduling algorithm which may be performed by the management server or network nodes of FIGs. 3 and 9. The method of FIG. 26 takes advantage of the original content playback speed and the aforementioned "catch-up period" T_{cp} . Again, T_{cp} comprises a predefined threshold that defines the maximum acceptable delay for recovery data content playback per the system policies and/or user preference. In one case, the recovery latency T_{cc} is less than or equal to the catch-up period ($T_{cc}(n) \leq T_{cp}$ for $n \in [1, N]$).

[00124] Steps 2610, 2615, 2620, 2625, 2630, 2635, 2640, 2645, 2650, 2655, 2660, 2665, 2670, 2675, 2680 and 2685 are respectively equivalent to steps 1310, 1315, 1320, 1325, 1330, 1335, 1340, 1345, 1350, 1355, 1360, 1365, 1370, 1375, 1380 and 1385 of FIG. 15.

[00125] The difference between the algorithm of FIG. 15 and that of FIG. 26 is that after a determination is made to cache $X(\tilde{n})$, a determination is made at 1340 or 1365

(and n incremented at 1370), a determination is made as to whether to cache transformed recovery data at 1382 or optimized recovery data at 1392. If transformed recovery data is to be cached, then at 1384, trans-formatting occurs to create a transformed recovery data segment/packet $W(\tilde{n})$ which is cached at 1386. If playback optimized recovery data is to be cached, then at 1394, intelligent processing occurs and a playback optimized recovery segment/packet $X'(\tilde{n})$ is cached at 1396. The method then returns to 1345 and proceeds to loop through step 1315 until the conference ends.

[00126] FIG. 27 illustrates the timing of segments/packets transmitted between a source participant device and a receiver device, and receiver device playback of transformed data packets $W(n)$. As illustrated in FIG. 27, the transformed recovery data packets $W(n)$ are transmitted and arriving out of sequence. Line 2710 shows data packets $X(1) - X(n)$ transmitted from a source device over time. Line 2720 shows arrival at a first participant device, for example device 312 in FIG. 3, without packet loss and at a slightly later time relative to transmission due to network delay. Line 2730 illustrates packets arriving at a second participant device with packets $X(4) - X(7)$ being lost in transmission at 2850. In accordance with the embodiment of FIG. 22, transformed recovery data content in the form of segment/packets $W(4) - W(7)$ are received at a time between segments/packets $X(8)$ and $X(n)$. The receiver participant device playback rendering of segments/packets $W(4) - W(7)$ can begin immediately at 2780, since the transformed data can be overlaid on the live meeting data (as in FIG. 22) or presented in another format.

[00127] FIG. 28 illustrates the timing of segments/packets transmitted between a source participant device and a receiver device, along with the latencies introduced using the transformed recovery data method of FIG. 21. FIG. 28 illustrates arriving packets on line 2810, rendering of data on line 2820 and acknowledgements in the cloud at line 2830. In FIG. 28, as in FIG. 27, packets sequence numbers 4 – 7 ($X(4) - X(7)$) are lost and transformed recovery data packets sequence numbers 4 – 7 arrive at time τ_v4 . The playback start time begins on arrival of transformed packet sequence number 4 ($W(4)$) with the latency reduced since the transformed data is overlaid with

the real-time data of real-time packets 8 – 11. As shown in FIG. 28, receiver playback of transformed data may occur at 2840, with a recovery buffer latency T_{cb} and a relatively small recovery catch-up latency T_{cc} since recovery data can be displayed with real-time meeting data (packet sequence numbers 8 – 20).

[00128] FIG. 29 illustrates an example of a playback optimized segment/packet of data $X'(n)$. FIG. 29 illustrates an example of an audio waveform in an original segment. As illustrated therein through cropping and removal of silent or near silent segments 2902, 2904, the audio data is significantly reduced. Similar optimization can be applied to video data and can be based on audio data. For example, where video data is synced to audio data and silent periods exist in the audio data, if the video data contains no meeting information, intelligent processing can remove portions of video data associated with silent periods in the audio. In another example, where video data is merely a slide presentation by a source, a playback optimized packet may capture the slide image rather than including video of the slide, thereby substantially reducing the recovery segment/packet size.

[00129] FIG. 30 illustrates the timing of segments/packets transmitted between a source participant device and a receiver device, with receiver device playback of playback optimized data segments/packets $X'(n)$. As illustrated in FIG. 31, $X(4)$ is lost. Once one or more packets are lost, playback optimized data segments/packets $X'(n)$ (in this case playback optimized data segments/packets $X'(4) - X'(8)$) are received by the device which has suffered a lost segments/packet. Line 3010 shows data segments/packets $X(1) - X(n)$ transmitted from a source device over time. Line 3020 illustrates segments/packets arriving at a participant device with packet $X(4)$ being lost in transmission at 3050. In accordance with the embodiment of FIG. 23, playback optimized data segments/packets $X'(4) - X'(8)$ are received at a time after segment/packet $X(4)$ and rendering (line 3030) occurs at a normal playback speed until segment/packet $X(9)$, which is at a catch-up point allowing the receiver device to be in sync with other meeting participants. Rendering of optimized packets $X'(4) - X'(8)$ in this embodiment is illustrated as occurring at normal speed, but a participant may have a different meeting experience during the recovery period, as choppy audio or visual data, due to the removal of quiet portions of the stream.

[00130] FIG. 31 illustrates the relative timing for one example of how client device data queueing and playback can correct for lost data in the present technology. In FIG. 31, illustrates the timing of segments/packets transmitted between a source participant device and a receiver device, along with the latencies introduced using the transformed recovery data method of FIG. 22. FIG. 31 illustrates arriving packets on line 3110, rendering of data on line 3120 and acknowledgements in the cloud at line 3130. In FIG. 31, packets sequence numbers 4 – 7 ($X(4) - X(7)$) are lost and transformed recovery data packets sequence numbers 4' – j arrive at time $\tau_v(4)$. The playback start time begins on arrival of playback optimized sequence number 4' ($X'(4)$) with the latency being slightly longer than with transformed data since the optimized must catch-up with the real-time data of real-time packets. The playback start time begins on arrival of playback optimized data packets $X'(t)$ with the catch-up latency reduced since the transformed data is overlaid with the real-time data of packets 8 – 11.

CLIENT ADAPTATION FOR REAL-TIME MEETING DATA RECOVERY

[00131] Another aspect of the technology includes real-time meeting recovery using client-device controlled techniques. In the aforementioned real-time data recovery embodiments, rendering of recovery data has generally taken place at the same rate or speed as real-time data segments/packets. In embodiments, additional control over recovery rendering on participant devices may be utilized. In general, in addition to recovery rendering playback schemes that comprise playback at normal speed, accelerated playback with real-time recovery, and/or jump forward real-time recovery at the receiver device, may also be utilized.

[00132] FIG. 32 illustrates a general method performed at a participant (receiving) device for implementing participant device-controlled data recovery. In the below embodiments, $X^*(n)$ designates a processed segment/packet in accordance with any of the embodiments herein (such as a transformed segment/packet $W(n)$ or a playback optimized segment/packet $X'(n)$). At 3210, the meeting real-time content begins and participants in the meeting share meeting information as discussed herein. Once the

real-time online meeting has begun at 3210, the method determines at 3215 whether any segments/packets have not been delivered to the device by checking segment/packet sequence numbers and a segment/packet timeout. At 3215, if segments/packets are continually received within the timeout period, then the real-time meeting content is continually rendered on participant devices at 3225. If a segment/packet is lost at 3215, a decision is made at 3220 as to whether or not to initiate the real-time recovery on the participant device. If not, the participant device continues rendering real-time live meeting content at 3225 without those segments/packets which were dropped. If real-time content recovery is initiated at 3220, then at 3240, a participant device-controlled recovery playback scheme is selected. Examples of playback methods are illustrated in FIGs. 33 and 34. The decision at 3240 may be based on predefined parameters including a system policy 465 which may be defined by the meeting service provider, user preferences 470, and resource availability 475. At 3260, participant device client recovery is performed until a catch-up point is reached at which point normal rendering of the live-meeting stream occurs at 3270.

[00133] In a first method, the participant receiver device can playback the n th segment, $X(n)$ or $X^*(n)$ and all following segments $X(n+1)$, $X(n+2)$, etc. or $X^*(n+1)$, $X^*(n+2)$, etc. at normal rendering rate or speed, but with a delay τ_D . The receiver participant device can then catch-up to the real-time meeting activity with the rest of the meeting participants at a catch-up point, such as during audio pause, meeting break, speaker change, or other detected point. In embodiments, the receiver participant device may jump forward, thus bypassing certain content segments/packets that are not of interest to the participant. Jumps conducted automatically per user preferences or manually by the participant. The catch-up point may be during or after the jump forward operation.

[00134] In another participant device recovery method, the receiver participant device can playback the n th segment $X(n)$ or $X^*(n)$ and a number of segments "K" following $X(n)$ or $X^*(n)$ -- $X(n+1)$, $X(n+2)$, $X(n+K)$ or $X^*(n+1)$, $X^*(n+2)$, $X^*(n+K)$ -- at an accelerated rendering rate that is greater (or faster) than the normal

rendering rate. In this embodiment, the participant device can synchronize data with the rest of the meeting participants after K segments. Jumping forward may also be performed in this embodiment. Assuming playback speed is set to be SRp times of the original speed Sp , with $SRp > Sp$, based on the system specification and/or user preference, then then $K = \frac{Sp}{SRp - Sp}$, e.g. $SRp = 1.25 \times Sp$, then $K = 4$, for single segment/packet loss and $K = \frac{Sp \times H}{SRp - Sp}$ for multiple segments/packets loss (where H is the number of lost segments/packets).

[00135] This participant side recovery has the advantage of requiring only minimum additional computing resources and can be used with any of the recovery data formats and delivery schemes discussed herein (original, transformed and playback-optimized).

[00136] FIG. 33 illustrates one embodiment of step 3260 of FIG. 32. At 3320, an initial determination is made as to whether the method will use a playback skip (jump) based on user preferences and system settings. If so, then at 3330, the packet sequence number " n " and the number of segments " k " following $X(n)$ are incremented, and the number of lost segments/packets " H " is decremented. Once the number of segments following $X(n)$ and the number of lost packets is the same, normal streaming resumes at 3380. If no skip is made at 3320, then at 3350, playback occurs at original until a catch-up point is reached at 3360. Until the catch-up point is reached at 3360, the packet sequence number " n " and the number of segments/packets " k " following $X(n)$ are incremented as playback rendering occurs.

[00137] FIG. 34 illustrates another embodiment of step 3360 of FIG. 32. At 3420, an initial determination is made as to whether the method will use a playback skip (jump) based on user preferences and system settings. If so, then at 3430, the segments/packets sequence number " n " and the number of segments/packets " k " following $X(n)$ are incremented, and the number of lost segments/packets " H " is decremented. Once the number of segments/packets following $X(n)$ and the number of lost segments/packets are the same, normal streaming resumes at 3480. If no skip is made at 3320, then at 3450, playback occurs at an accelerated (second/faster)

speed SRp until a $K = \frac{Sp}{SRp - Sp}$ is reached at 3460. Until the $K = \frac{Sp}{SRp - Sp}$ at 3460, the segments/packets sequence number " n " and the number of segments/packets " k " following $X(n)$ are incremented as playback rendering occurs.

[00138] In embodiments, participant side recovery may be used with both local caching and/or caching at one or more of the network devices in the network environment. FIG. 35 illustrates a method for participant device recovery with network device caching. At 3510, a determination is made as to whether the full catch-up content is present in a local cache. If so, then catch-up content playback can begin. If not, at 3520, a service request is sent to a network device for the recovery content. Beginning with sequence number $n=1$ at 3540, the method first determines whether $X(n)$ is in a local cache and if so, sends $X(n)$ to the playback queue at 3570. If not, $X(n)$ is fetched from the network device at 3560 and sent to the playback queue at 3570. If the recovery content is not completed, the sequence number is incremented at 3590 and the method continues until recovery content is fully played back at 3580.

[00139] FIG. 36 illustrates a comparison of the relative timing of segments/packets transmitted between a source participant device and two receiving participant devices Ca and Cb. Line 3610 shows data packets $X(1) - X(n)$ transmitted from a source device over time. Line 3620 shows arrival at a first participant device Cb, for example device 312 in FIG. 3, without packet loss and at a slightly later time relative to transmission due to network delay. Lines 3630 and 3640 compare segments/packets arriving at a second participant device Ca with packet $X(4)$ being lost in transmission. At line 3630, recovery data segment/packet $X(4)$ arrives after arrival of real-time data packet $X(5)$, while at line 3640, the segments/packets are shown arriving in sequence. At line 3650, the rendering order of participant device Cb is illustrated and follows the sequence of segment/packet receipt at 3620 (following network and buffer latency).

[00140] Lines 3660 and 3670 compare the rendering order and speed of the embodiments of FIG. 36 with a jump forward and no-jump forward, respectively. At 3660, participant device Ca utilizes jump forward to skip segments/packets $X(6)$ and $X(7)$ to quickly catch-up in speed to segment/packet $X(8)$ at catch-up point 3680. It will be noted that participant Ca's rendering is now in sync with that of participant device

Cb in line 3650. Line 3670 illustrates play without a jump forward, and participant device Ca will not re-sync rendering with participant device Cb until a pause or break occurs at 3690.

[00141] FIG. 37 illustrates the relative timing of segments/packets transmitted between a source participant device and two client devices Ca and Cb where the rendering method of FIG. 34 is used (i.e. accelerated playback). Lines 3910, 3920 and 3930 are equivalent to the segment/packet transmission and delivery representations in FIG. 39. Line 3740 illustrates participant device Cb using no accelerated playback or jump forwarding. Line 3750 illustrated device Ca using accelerated playback rate of 1.25 times normal. For one lost packet, five playback optimized packets are utilized until device Ca is at the same synchronization as device Cb. Accelerated playback occurs until a catch-up point 3720 at which point playback is synchronized with other participant devices.

[00142] It should be understood that the embodiments of FIGs. 36 and 37 may be combined. Thus, in one embodiment, both skipping and accelerated playback may be used to reach a catch-up point.

REAL-TIME MEETING DATA RECOVERY AFTER PROACTIVE PARTICIPANT INTERRUPTION

[00143] The real-time content recovery techniques discussed herein have thus far focused on content recovery cause by data loss or corruption due to network issues. The aforementioned recovery techniques can also be applied based on proactive actions of a participant at a receiving device, allowing participants to proactively pause or take a break from a real-time online meeting and later recover missed content in multi-person real-time online meetings.

[00144] In one aspect, a proactive content recovery method is identical to that of FIG. 4 except that step 425 (segment/packet loss detection by timeout) and 435 (initiate real-time recovery) are proactively initiated by a participant.

[00145] FIG. 38 is a general overview flowchart illustrating various embodiments of proactive initiation of real-time content recovery in an online meeting in combination with the aforementioned recovery schemes, including recovery using original format recovery data, playback optimized recovery data, transformed recovery data, and participant device compensating real-time content recovery embodiments. Once a participant initiates a break by generating a rendering interrupt, real-time data may be cached as recovery data (in any of the formats disclosed herein) on the device itself or retrieved from caches on one or more network devices to initiate a recovery of real-time meeting content. In embodiments, a single sub-queue on the client device may be used to queue data, or multiple sub queues may be utilized. Queuing may also be split between local caches and caches on network devices. Packets/segments can be placed into two or more sub queues based on sequence number, type of the segments/packets, and delivery deadlines. In one embodiment, recovery packets/segments are placed in a separate sub queue from original data packets/segments sub queue. As in other embodiments, priorities of the sub queues are defined based on application specific or system policies and rules, as well as user preferences. Intelligent queueing and caching can also be used where queueing and caching is based on system policies and user preferences.

[00146] With reference to FIG. 38, at 3802, the user will initiate a proactive break in the real-time online meeting at their participant device. A proactive break or pause by the user may generate a rendering interrupt at the user's participant processing device. The break ends when the user initiates a restart, with the time period between the interrupt and the re-start comprising a pause. At 3804, data rendering pauses on the client device. The last playback segment/packet sequence number "L" is recorded at 3806. At 3808, a determination is made as to whether not recovery processing should be enabled. If it is not enabled, then the current session is terminated at 3810 and the user will need to restart or rejoin the meeting.

[00147] If recovery is initiated at 3808, the available resources are estimated at 3812. At 3814, the total number of available sub queues J will be set equal to the total recovery packet sequence number j, the buffer latency T_{cb} is set equal to 0 and the latency gain due to the removal of all null segments/packets T_{cg} is also set to 0. At

3816, the number of sub-queues would be incremented by one. At 3818, the type of recovery scheme which will be used will be selected from between the various embodiments described herein. If original data format recovery is selected, then at 3828, the method will cache the next data packet $X(j)$ and if the break has not ended at 3830, the buffer latency will be increased at 3832, and the number of buffers incremented by one at 3834. If transformed data recovery is utilized, then at 3820 the method will request the transformed data packet $W(j)$ and cache it, and if the break has not ended at 3822, the buffer latency will be increased at 3824 and number of buffers incremented by one at 3826. If intelligent data recovery with optimized playback packets is selected, then at 3836 if segment/packet $X(j)$ is not null, the system will cache $X(j)$ and continue to check whether the break is ended at 3842. If the break has not ended at 3842, then the then the buffer latency T_{cb} will be incremented by the cache $t(j)$ latency at 3844. If $X(j)$ is null at 3836, the buffer latency T_{cb} will be incremented by the cache $t(j)$ latency at 3840. When the break ends at any of steps 3842, 3830 and 3822 the data rendering will begin at 3848, progressing packets $X(j)$ in order, incrementing the cached packed number j at 3850 until all packets are removed from the cache.

[00148] FIG. 39 is a general overview flowchart illustrating local and network caching which may be used with the proactive implemented content recovery. In a first caching scheme, where a local cache contains all of the local caching a receiving participant device, upon receiving 'break' initiation instruction, shall cache the subsequent content stream $X(n), X(n+1) \dots$, until 'end of break' instruction received or until a predefined deadline is reached. In a second scheme, edge caching or joint edge/device caching is used. The receiver device, upon receiving 'break' initiation instruction, shall request edge server/network node to cache all or shall work jointly with the edge server/network node to cache all subsequent content stream $X(n), X(n+1) \dots$, until 'end of break' instruction received or until a predefined deadline is reached. The estimated cache size needed for break content buffering (Ch_{brk}) is thus $Ch_{brk} = \sum_{n=1}^{\Omega_{brk}} \beta(n)$.

[00149] At 3902, a break mode is initiated by a user. If proactive recovery is initiated, proactive recovery begins at 3906. If proactive recovery is not initiated at 3904, then

at 3908, the cache size for content buffering Ch_{brk} is estimated at 3908. The sequence number is set to one at 3910 and at 3912 a determination is made as to whether the estimated local available cache size Ch_{dev} is greater than or equal to the estimated needed buffering size Ch_{brk} . If so, then a determination of whether local caching will be used at 3914 is made. If so, then at 3918, the system caches real-time data packets at 3918 as long as the break remains active at 3916. The sequence number is incremented at 3920 and when the break ends at 3916, playback or rendering is resumed at 3922. At the local available cache size is not greater than or equal to the estimated needed buffering size of at 3912, or a local cache is determined not to be used at 3914, joint edge device caching is used at 3924. As long as the break does not end at 3926, a check is made at 3928 to determine whether the local available cache size is full. If Ch_{dev} is not full, the real-time media data packet is cached locally at 3932, the sequence number incremented at 3934 and the system loops back to step 3926. If the local cache is full at 3920, then the real-time data is cached in one of the network devices. In one embodiment, caching occurs at 3920 at a device closer to the network location where the participant device performing the method of FIG. 39 is operating. When the break ends at 3926, rendering resumes at 3922.

[00150] FIG 40 illustrates two types of content rendering when a proactive, user-initiated break is initiated, based on the caching method illustrated in FIG. 39. It should be noted that where all original format recovery data is cached locally at a participant device, the participant device does not need to send a service request to other devices in the network. In embodiments where both local and network caching is utilized, a service request similar to the service request issued by device 510 in FIG. 5 is sent to network devices handling recovery data for the participant device. In other embodiments, even where local caching is utilized, the receiving participant device may notify other devices of the proactive break initiated to let other participants know that the receiving participant device is temporarily paused.

[00151] Line 4010 illustrates segments/packets (sequence numbers only) being received at a participant device. A break 4012 is initiated when segment/packet sequence number 3 is received and ends at sequence number 7. Original format

meeting data recovery (in accordance with the methods discussed above at FIG. 20, for example,) is initiated at the end of the break.

[00152] A first rendering scheme illustrated at line 4030 assumes all recovery data is cached locally and thus a local buffer is used to provide recovery data in its original format until segments/packets are rendered and the rendering reaches catch-up point τ_{cp} at 4027. Buffer latency is introduced on the receiving device along with a recovery buffer latency T_{cb} of $\tau_v(4)$ minus $\tau_u(4)$ so that sequence numbers $4 - (k-1)$ (where “ k ” is the next real-time packet rendered in sequence after the catch-up point τ_{cp}) can be rendered in sequence.

[00153] In a second rendering scheme illustrated at line 4040, local and network caches cooperate to provide recovery data in its original format until segments/packets are rendered and the rendering reaches catch-up point τ_{cp} at 4028. As illustrated at line 4040, any number of cached original format data segments/packets having sequence number “ j ” will be rendered along with local packet. Again, playback buffer latency is introduced on the receiving device along with a recovery buffer latency T_{cb} of $\tau_v(4)$ minus $\tau_u(4)$ so that sequence numbers $4 - j - (k-1)$ are rendered in sequence. As shown at 4042, network devices receive packet receipt acknowledgment in sequence with the sequence numbers of line 4010.

[00154] FIG. 41 illustrates client device content queuing and playback for a proactive break when transformed recovery data is utilized. Where transformed recovery data is utilized with a proactive break, the receiving participant device may be configured to prepare the transformed recovery data locally. In other embodiments, the transformed recovery data is created at one or more network nodes. Thus, the receiving device will issue a service request when the proactive break is started. FIG. 41 illustrates an embodiment where the transformed recovery data and playback optimized recovery data is created at one or more network nodes.

[00155] Line 4110 illustrates transformed packets (sequence numbers only) being received at a participant device. As shown therein, transformed data 4150 is received after a break 4112 ends at sequence number 7 and after real-time meeting packets 8

– 10 (in transmission from a source device) are received. Transformed recovery data covering the break time period and number of real-time segments/packets not rendered at the participant device are determined and received from one or more network devices (in this embodiment). Transformed data begins rendering and display data over real-time segments/packets in sync with other participants (segment/packet sequence numbers $8 - n$) as the real-time segments/packets are rendered at τ_v . As shown at 4142, network devices receive packet receipt acknowledgment in sequence with the sequence numbers of line 4110 followed by acks for the recovery data.

[00156] FIG. 42 illustrates client device content queuing and playback for a proactive break when playback optimized recovery data is utilized. Where playback optimized recovery data is utilized with a proactive break, the receiving participant device may be configured to prepare the playback optimized recovery data locally. In other embodiments, the playback optimized recovery data is created at one or more network nodes. Thus, the receiving device will issue a service request when the proactive break is started where playback optimized data is cached at other devices. FIG. 42 illustrates an embodiment where the transformed recovery data and playback optimized recovery data is created at one or more network nodes.

[00157] Line 4210 illustrates playback optimized segments/packets (sequence numbers only) being received at a participant device. As shown therein, playback optimized data 4250 is received after a break 4212 ends at sequence number 7 and after real-time meeting packets 8 – 10 (which were already in transmission from a source device) are received. Playback optimized recovery data covering the break time period and a number of real-time segments/packets not rendered at the participant device are determined and received from one or more network devices (in this embodiment) at 4250. Playback optimized data begins rendering at $\tau_v(4)$ and until a catch-up point τ_{cp} is reached, with segment/packet number “j” indicating that the playback optimized data packets may be any number of packets received from network caches. As shown at 4242, network devices receive packet receipt acknowledgment in sequence with the sequence numbers of line 4110 followed by acks for the playback optimized recovery data.

[00158] In other embodiments, caching or recovery of any of the above types of data can be controlled using machine learning. For recurring live meetings, historical data available from previous meeting can be used to train and predict cache availability and bandwidth available for regular participants using machine learning algorithms.

[00159] In other embodiments, the relative amount of data cached at each of the cloud, edge and client devices can be distributed differently, according to one or more various caching distribution algorithms and/or user preferences.

[00160] In other embodiments, trans-formatted catch-up data may be utilized simultaneously with real-time data rendering when a proactive user pause is initiated at a client device.

[00161] FIG. 43 illustrates an interface 4300 of an online conference application showing an example of simultaneously rendered live audio/visual information and recovery data. Interface 4300 includes a live data presentation window 4302 where live content from a presenter, or a focused attendee (a video, picture, or other representation) is displayed. Interface 4300 includes windows 4330 showing other connected attendees of the real-time meeting. In this example, the presenter window 4302 is showing an attendee but may also include text, video or shared screen information. In addition, a recovery data display window 4304 allows simultaneous presentation of recovery data. In embodiments, the simultaneous presentation window 4304 may display the original content segments (missing original $X(n)$ content) in a silent mode with audio content transcribed as text, or summarized for display. The simultaneous presentation window may display transformed recovery data (in accordance with any of the formats discussed herein) or playback optimized recovery data, or any of the types of recovery data described herein. In addition, the placement of the windows may differ in various embodiments.

[00162] FIG. 44 illustrates the timing of segments/packets transmitted between a source participant device and a receiver device when a proactive break is initiated and the playback interface of FIG. 43 is utilized. Although FIG. 44 illustrates playback of transformed data segments/packets $W(n)$, any type of recovery data may be utilized in this playback illustration. Line 4410 illustrates live data packets transmitted from a

source, line 4420 illustrates the segments/packets arriving at a receiver and line 4440 illustrates receiver rendering (or playback) of the live and recovery data segments/packets. As illustrated in FIG. 44, a user initiates a break start at 4430, one or more of the client device, a network node or the meeting server may begin caching the live data segments/packets and transforming the live data to recovery data. In this example, the transformed recovery data packets $W(n)$ are transformed and illustrated in sequence. In accordance with the embodiment of FIG. 43, transformed recovery data content in the form of segment/packets $W(4) - W(7)$ are rendered at a time 4480 simultaneously with segments/packets $X(8)$ and $X(n)$. The receiver participant device playback can begin immediately at 4480, since the transformed (or cached live) data can be presented simultaneously with the live data.

[00163] FIG. 45 illustrates another embodiment of proactive user-initiated interruption of a live meeting or presentation using a content-aware break or pause. In this embodiment, a network node or server (or the client itself) allows a meeting participant to take a proactive break at a time when the participant specifies a certain type of content that the participant is less interested in. This allows the participant to catch up with the live meeting or presentation more quickly in some cases. At 4510, a meeting participant initiates proactive break notification. The break notification may be for a period N_p when specific content C_{sp} or a specific type of content is to be delivered. One example of a specific type of content is when the presenter is discussing a type of content the participant is not interested in, or is presenting a publicly available video clip. In another example, during a conference with multiple speakers delivering multiple talks one after another, a meeting participant may be specifically interested in some talks but not others.

[00164] At 4515, a meeting participant may identify a content type or specific content during which presentation the participant wishes to take a break. At 4520, in one embodiment, the participant's client device may send a notification to the meeting server or a network node implementing content aware interruption. At 4525, the node or server will determine whether the specific content or content type is detected and when detected, will notify the client at 4530. In one embodiment, the server or node detects the start of an N_p , or predicts an N_p , will start soon at a specific time. Thus the

notification at 4520 may be to let the client know that it can take a break immediately or at a specific future time. At 4545, the server can cache the live content as recovery content in its original format or prepare transformed, optimized, or other forms of recovery content of any of the types described herein and cache such content for use in recovery after the proactive break.

[00165] After the client is notified at 4530, the participant can choose to begin the break immediately at 4535 or start the break at a predicted time at 4540. When a participant ends the break or when the rendering period for the identified content expires at 4550, recovery content is requested at 4555 and content is forwarded from the node or server to the client for rendering at 4560. In another embodiment, the server may choose to intelligently summarize the content and forward this recovery data to the client.

[00166] In another embodiment, the method of FIG 45 may be performed entirely on the participant device, such that notification 4520, 4530 need not occur, and content detection at 4525 and recovery content caching and (optionally) transformation 4545 may occur on the client device.

[00167] FIG. 46 is a block diagram of a network processing device that can be used to implement various embodiments. Specific network devices may utilize all of the components shown, or only a subset of the components, and levels of integration may vary from device to device. Furthermore, the network device 4600 may contain multiple instances of a component, such as multiple processing units, processors, memories, transmitters, receivers, etc. The network device 4600 may comprise a processing unit 4601 equipped with one or more input/output devices, such as network interfaces, storage interfaces, and the like. The processing unit 4601 may include a central processing unit (CPU) 4610, a memory 4620, a mass storage device 4630, and an I/O interface 4660 connected to a bus 4670. The bus 4670 may be one or more of any type of several bus architectures including a memory bus or memory controller, a peripheral bus, or the like. A network interface 4650 enables the network processing device to communicate over a network 4680 with other processing devices such as those described herein.

[00168] The CPU 4610 may comprise any type of electronic data processor. The memory 4620 may comprise any type of system memory such as static random-access memory (SRAM), dynamic random-access memory (DRAM), synchronous DRAM (SDRAM), read-only memory (ROM), a combination thereof, or the like. In an embodiment, the memory 4620 may include ROM for use at boot-up, and DRAM for program and data storage for use while executing programs. In embodiments, the memory 4620 is non-transitory. In one embodiment, the memory 4620 includes computer readable instructions that are executed by the processor(s) 1320 to implement embodiments of the disclosed technology, including the live meeting application 4625A which may itself include a rendering engine 4625b, sequence number data store 4625c, live stream data buffer and cache 4625d, a recovery data cache 4625e and a recovery content service application 4625f. The functions of the meeting application 4625a, live stream data buffer and cache 4625d, a recovery data cache 4625e and a recovery content service application 4625f are described herein in various flowcharts and Figures.

[00169] The mass storage device 4630 may comprise any type of storage device configured to store data, programs, and other information and to make the data, programs, and other information accessible via the bus 4670. The mass storage device 4630 may comprise, for example, one or more of a solid-state drive, hard disk drive, a magnetic disk drive, an optical disk drive, or the like.

[00170] FIG. 47 is a block diagram of a network processing device that can be used to implement various embodiments of a meeting server 340 or network node 320 or 330. Specific network devices may utilize all of the components shown, or only a subset of the components, and levels of integration may vary from device to device. In FIG. 46, like numbers represent like parts with respect to those of FIG. 46. In one embodiment, the memory 4620 includes the live meeting service application 4615A including and sequence number data 4615c. The memory may also include the recovery content service application 4615b which includes trans-formatting engine 4615e and intelligent recovery data generator 4615f. The recovery content service application 4615b responds to service requests generated by participant devices to provide recovery data under the particular embodiments discussed herein. The trans-

formatting engine 4615e generates transformed recovery data as described herein and the intelligent recovery data generator generates playback optimized recovery data as described herein.

[00171] FIG. 48 is a block diagram illustrating examples of details of a network device, or node, such as those shown in the network of FIG. 3. A node 4800 may comprise a router, switch, server, or other network device, according to an embodiment. The node 4800 can correspond to one of the nodes 320a- 320d, 330a – 330d. The router or other network node 4800 can be configured to implement or support embodiments of the technology disclosed herein. The node 4800 may comprise a number of receiving input/output (I/O) ports 4810, a receiver 4812 for receiving packets, a number of transmitting I/O ports 4830 and a transmitter 4832 for forwarding packets. Although shown separated into an input section and an output section in FIG. 48, often these will be I/O ports 4810 and 4830 that are used for both down-stream and up-stream transfers and the receiver 4812 and transmitter 4832 will be transceivers. Together I/O ports 4810, receiver 4812, I/O ports 4830, and transmitter 4832 can be collectively referred to as a network interface that is configured to receive and transmit packets over a network.

[00172] The node 4800 can also include a processor 4820 that can be formed of one or more processing circuits and a memory or storage section 4822. The storage 4822 can be variously embodied based on available memory technologies and in this embodiment and is shown to have recovery data cache 4870, which could be formed from a volatile RAM memory such as SRAM or DRAM, and long-term storage 4826, which can be formed of non-volatile memory such as flash NAND memory or other memory technologies.

[00173] Storage 4822 can be used for storing both data and instructions for implementing the real-time recovery data techniques herein. In particular, instructions causing the processor 4820 to perform the functions of caching recovery data in original data formats, and/or transforming recovery data into the different data formats discussed herein and caching recovery data in different data formats.

[00174] Other elements on node 4800 can include the programmable content forwarding plane 4828. Depending on the embodiment, the programmable content forwarding plane 4828 can be part of the more general processing elements of the processor 4820 or a dedicated portion of the processing circuitry.

[00175] More specifically, the processor(s) 4820, including the programmable content forwarding plane 4828, can be configured to implement embodiments of the disclosed technology described below. In accordance with certain embodiments, the storage 4822 stores computer readable instructions that are executed by the processor(s) 4820 to implement embodiments of the disclosed technology. It would also be possible for embodiments of the disclosed technology described below to be implemented, at least partially, using hardware logic components, such as, but not limited to, Field-programmable Gate Arrays (FPGAs), Application-specific Integrated Circuits (ASICs), Application-Specific Standard Products (ASSPs), System-on-a-chip systems (SOCs), Complex Programmable Logic Devices (CPLDs), special purpose computers, etc.

[00176] FIG. 49 illustrates one embodiment of a network packet implementation for enabling real-time data recovery. The network packet of FIG. 49 may be used to communicate to the participant and network devices disclosed herein the real-time meeting data recovery techniques which are to be utilized. In the example of FIG. 49, two or more bits may be used to indicate up to four different schemes of the real-time content recovery techniques described herein. As is generally well known, the TCP/IP protocol stack 4910 commonly used in internet communications includes an IP Header, a TCP/UDP header, an application protocol header, and a data payload. The custom application layer provided between the TCP/UDP header, and the data payload may include an identifier in a reserved portion of the application layer protocol header. Bit 0 may be used to indicate that real-time data recovery is to be used (for example, data "0" for no recovery and "1" for recovery). Bits 1 and 2 may identify the types of recovery in use, for example proactive, original format data, transformed data or intelligent, playback optimized. This identifier may be read by any of the devices in the network environment to indicate the type of real-time content recovery in use in

the system. Each device can then act accordingly based on the configuration of the network environment.

[00177] For purposes of this document, it should be noted that the dimensions of the various features depicted in the figures may not necessarily be drawn to scale.

[00178] For purposes of this document, reference in the specification to “an embodiment,” “one embodiment,” “some embodiments,” or “another embodiment” may be used to describe different embodiments or the same embodiment.

[00179] For purposes of this document, a connection may be a direct connection or an indirect connection (e.g., via one or more other parts). In some cases, when an element is referred to as being connected or coupled to another element, the element may be directly connected to the other element or indirectly connected to the other element via intervening elements. When an element is referred to as being directly connected to another element, then there are no intervening elements between the element and the other element. Two devices are “in communication” if they are directly or indirectly connected so that they can communicate electronic signals between them.

[00180] Although the present disclosure has been described with reference to specific features and embodiments thereof, it is evident that various modifications and combinations can be made thereto without departing from scope of the disclosure. The specification and drawings are, accordingly, to be regarded simply as an illustration of the disclosure as defined by the appended claims, and are contemplated to cover any and all modifications, variations, combinations, or equivalents that fall within the scope of the present disclosure.

[00181] The technology described herein can be implemented using hardware, software, or a combination of both hardware and software. The software used is stored on one or more of the processor readable storage devices described above to program one or more of the processors to perform the functions described herein. The processor readable storage devices can include computer readable media such as volatile and non-volatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer

readable storage media and communication media. Computer readable storage media may be implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Examples of computer readable storage media include RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information, and which can be accessed by a computer. A computer readable medium or media does (do) not include propagated, modulated, or transitory signals.

[00182] Communication media typically embodies computer readable instructions, data structures, program modules or other data in a propagated, modulated, or transitory data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term “modulated data signal” means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as RF and other wireless media. Combinations of any of the above are also included within the scope of computer readable media.

[00183] In alternative embodiments, some or all of the software can be replaced by dedicated hardware logic components. For example, and without limitation, illustrative types of hardware logic components that can be used include Field-programmable Gate Arrays (FPGAs), Application-specific Integrated Circuits (ASICs), Application-specific Standard Products (ASSPs), System-on-a-chip systems (SOCs), Complex Programmable Logic Devices (CPLDs), special purpose computers, etc. In one embodiment, software (stored on a storage device) implementing one or more embodiments is used to program one or more processors. The one or more processors can be in communication with one or more computer readable media/storage devices, peripherals and/or communication interfaces.

[00184] It is understood that the present subject matter may be embodied in many different forms and should not be construed as being limited to the embodiments set

forth herein. Rather, these embodiments are provided so that this subject matter will be thorough and complete and will fully convey the disclosure to those skilled in the art. Indeed, the subject matter is intended to cover alternatives, modifications, and equivalents of these embodiments, which are included within the scope and spirit of the subject matter as defined by the appended claims. Furthermore, in the following detailed description of the present subject matter, numerous specific details are set forth in order to provide a thorough understanding of the present subject matter. However, it will be clear to those of ordinary skill in the art that the present subject matter may be practiced without such specific details.

[00185] Aspects of the present disclosure are described herein with reference to flowchart illustrations and/or block diagrams of methods, apparatuses (systems) and computer program products according to embodiments of the disclosure. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general-purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable instruction execution apparatus, create a mechanism for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

[00186] The description of the present disclosure has been presented for purposes of illustration and description but is not intended to be exhaustive or limited to the disclosure in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art without departing from the scope and spirit of the disclosure. The aspects of the disclosure herein were chosen and described in order to best explain the principles of the disclosure and the practical application, and to enable others of ordinary skill in the art to understand the disclosure with various modifications as are suited to the particular use contemplated.

[00187] For purposes of this document, each process associated with the disclosed technology may be performed continuously and by one or more computing devices.

Each step in a process may be performed by the same or different computing devices as those used in other steps, and each step need not necessarily be performed by a single computing device.

[00188] Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.

CLAIMS

What is claimed is:

1. A computer implemented method of recovering real-time online meeting content which is lost, comprising:
 - transmitting and receiving real-time meeting data including meeting content to and from participant devices in a data format during a real-time online meeting;
 - determining that real-time meeting data transmitted in a sequence between devices during the real-time online meeting has been lost;
 - receiving real-time recovery data in a different data format, the real-time recovery data containing real-time online meeting content replacing meeting content in the lost real-time meeting data; and
 - rendering the real-time recovery data to replace the real-time meeting content in the lost real time meeting data.
2. The computer implemented method of claim 1, wherein the meeting content is in an audio/visual format and the different data format comprises a non-audio/visual format.
3. The computer implemented method of claim 1, wherein the meeting content is in an audio/visual format and the different data format comprises a playback rendering optimized format.
4. The computer implemented method of claims 1 and 2 wherein the rendering comprises rendering the real-time recovery data in at the same time as rendering real-time meeting data received after the lost real-time meeting data was lost.
5. The computer implemented method of claims 4 wherein the real-time recovery data comprises a text transcription of the audio/visual data and the rendering comprises rendering the text transcription of the audio/visual data the real-time

recovery data in at the same time as rendering real-time meeting data received after the lost real-time meeting data was lost.

6. The computer implemented method of claims 1 and 3 wherein the playback rendering optimized content comprises audio/visual content having a sub-set of both the audio and visual data in the lost real-time meeting data.

7. The computer implemented method of any of claims 1, 3 and 6 wherein the playback rendering optimized content comprises audio/visual content having a sub-set of less than all of audio data in the lost real-time meeting data.

8. The computer implemented method of claims 1, 3, 6 and 7 wherein the playback rendering optimized content comprises audio/visual content having null portions of the audio/visual content removed.

9. The computer implemented method of claim 1 through 8 wherein the real-time recovery data is received from a cache on a network device in a network environment.

10. A computer implemented method of recovering real-time meeting content which is lost during an online presentation, comprising:

transmitting and receiving real-time meeting data including real-time meeting content to and from participant devices in a data format during a real-time online meeting;

generating real-time content recovery data in a different data format, the real-time recovery data replacing meeting content in the real-time meeting data; and

forwarding the real-time recovery data to replace real-time meeting content which has been determined to be lost in communication to a host device, the real-time recovery data including content replacing the lost content to at least one participant device.

11. The computer implemented method of claim 10 wherein the meeting content is in an audio/visual format and the different data format a non-audio/visual format.

12. The computer implemented method of claim 11 wherein the meeting content is in an audio/visual format and the different data format comprises a playback rendering optimized format.

13. The computer implemented method of claims 10 and 11 wherein the generating comprises transforming the meeting content in the meeting data lost in communication to a host device into a transcription of the lost meeting data.

14. The computer implemented method of claims 10, 11 and 13 wherein generating comprises generating a text transcription of the audio/visual data which is forwarded for rendering at the same time as rendering real-time meeting data received after the lost real-time meeting data was lost.

15. The computer implemented method of claims 10 and 12 wherein generating comprises creating the playback rendering optimized content with audio/visual content which is a sub-set of both the audio and visual data in the lost real-time meeting data.

16. The computer implemented method of claims 10, 12 and 15 wherein the generating comprises generating playback rendering optimized content with only audio data, and which a sub-set of less than all of the audio/visual content in the lost real-time meeting data.

17. The computer implemented method of claims 10, 12, 15 and 16 wherein the playback rendering optimized content comprises audio/visual content having null portions of the audio/visual content removed.

18. A processing system in a network, comprising:
a storage medium comprising computer instructions;
one or more processors coupled to communicate with the storage medium,
wherein the one or more processors execute the instructions to cause the system to:

route real-time meeting data transmitted between a source device and at least one participant processing device during a real-time online meeting, the real-time meeting data including meeting content to and from participant devices in a data format during the real-time online meeting;

generate real-time recovery data in a different data format, the real-time recovery data replacing real-time meeting content in the real-time meeting data;

store the real-time recovery data; and

forward the real-time recovery data to replace real-time meeting content which has been determined to be lost in communication to at least one participant device, the real-time recovery data including content replacing the lost content to the at least one participant device.

19. The processing system of claim 18 wherein the meeting content is in an audio/visual format and the different data format a non-audio/visual format.

20. The processing system of claim 18 wherein the meeting content is in an audio/visual format and the different data format comprises a playback rendering optimized format.

21. The processing system of claims 18 and 19 wherein the one or more processors execute the instructions to generate by transforming the meeting content in the meeting data lost in communication to a host device into a transcription of the lost meeting data.

22. The processing system of claims 18, 19 and 21 wherein the real-time recovery data comprises a text transcription of the audio/visual data which is forwarded for rendering at the same time as rendering real-time meeting data received after the lost real-time meeting data was lost.

23. The processing system of claims 18 and 20 wherein the playback rendering optimized content comprises audio/visual content having a sub-set of both the audio and visual data in the lost real-time meeting data.

24. The processing system of claims 18, 20 and 23 wherein the playback rendering optimized content comprises audio/visual content having a sub-set of less than all of audio data in the lost real-time meeting data.
25. The processing system of claims 18, 20, 24 and 23 wherein the playback rendering optimized content comprises audio/visual content having null portions of the audio/visual content removed.
26. A user equipment device, comprising:
a storage medium comprising computer instructions;
one or more processors coupled to communicate with the storage medium,
wherein the one or more processors execute the instructions to cause the system to:
transmit and receive real-time meeting data including real-time live meeting content to and from participant devices in a data format during a real-time online meeting;
determine that real-time meeting data transmitted between devices during the real-time online meeting has been lost;
receive real-time recovery data in a different data format, the real-time recovery data replacing real-time live meeting content in the real-time meeting data which has been lost; and
rendering the real-time recovery data to replace the real-time meeting content in the lost real time meeting data.
27. The user equipment device of claim 26, wherein the meeting content is in an audio/visual format and the different data format a non-audio/visual format.
28. The user equipment device of claim 27, wherein the meeting content is in an audio/visual format and the different data format comprises a playback rendering optimized format.

29. The user equipment device of claims 26 and 27 wherein the rendering comprises forwarding the real-time recovery data for rendering at the same time as rendering real-time meeting data received after the lost real-time meeting data was lost.

30. The user equipment device of claim 29 wherein the real-time recovery data comprises a text transcription of the audio/visual data.

31. The user equipment device of claims 26 and 28 wherein the playback rendering optimized content comprises audio/visual content having a sub-set of both the audio and visual data in the lost real-time meeting data.

32. The user equipment device of any of claims 26, 28 and 31 wherein the playback rendering optimized content comprises audio/visual content having a sub-set of less than all of the audio data in the lost real-time meeting data.

33. The user equipment device of claims 26, 27, 31 and 32 wherein the playback rendering optimized content comprises audio/visual content having null portions of the audio/visual content removed.