Multimed Tools Appl
DOI 10.1007/s11042-010-0629-5

Next generation session management for 3D teleimmersive interactive environments

Klara Nahrstedt · Zhenyu Yang · Wanmin Wu · Ahsan Arefin · Raoul Rivas

Abstract In the recent past we have seen a boom of distributed interactive multimedia environments which use multiple correlated media sensors, multi-view displays, and advanced haptic-audio-visual user interfaces for teleimmersive gaming, business meetings and other collaborative activities. However, when we investigate the emerging teleimmersive environments closer, we realize that their overall session management, including end-to-end session setup, establishment and run-time management are not appropriate for the new demands that these environments present. These environments are cyber-physical rooms that demand (a) *large scale* of multi-sensory devices across geographically-distributed locations and (b) *interaction* with each other in *synchronous and real-time* manner. To deal with the new teleimmersive demands, we present a new session management design with (a) *session initiation protocol(s)* that understand media correlations, (b) *view-based multi-stream topology establishment* among multiple parties, (c) *efficient, light-weight and distributed session monitoring* with querying and debugging capabilities, (d) *flexible view-based session adaptation with efficient topology adjustments*, and (e) *light-weighted and consistent session tear-down protocols*. The presented design of the next generation session management

K. Nahrstedt · W. Wu · A. Arefin · R. Rivas
Department of Computer Science, University of Illinois at Urbana-Champaign, 201 N. Goodwin Avenue, Urban, IL 61801–2302, USA

K. Nahrstedt
e-mail: klara@cs.uiuc.edu

W. Wu
e-mail: wwu23@cs.uiuc.edu

A. Arefin
e-mail: marefin2@cs.uiuc.edu

R. Rivas
e-mail: trivas@uiuc.edu

Z. Yang (✉)
School of Computing and Information Sciences, Florida International University, 11200 SW 8th Street, Miami, FL 33199, USA
e-mail: yangz@cis.fiu.edu

Published online: 05 November 2010

protocols, services, algorithms and data structures is based on our extensive experiences with building 3D teleimmersive interactive systems, experimenting with high impact tele-immersive applications and deploying such environments at various venues.

Keywords Session management · 3D teleimmersive environments · Interactive multimedia systems

1 Introduction

In the recent past we have seen a boom of multi-sensory, multi-modal distributed interactive multimedia environments which are much more advanced and media-enriched than traditional audio-visual interactive systems such as Skype, NetMeeting, Yahoo Messenger, QQ and others [17, 19, 27, 43]. We are experiencing advanced collaborative interactive environments with multiple correlated media sensors within each room environment, multi-view displays, advanced haptic-audio-visual user interfaces such as 3D Teleimmersion environments (3DTI) at Illinois, Berkeley, UNC, University of Pennsylvania [8, 14, 20, 38] and other research laboratories, and 2D tele-presence rooms from HP and Cisco.

However, when we investigate these advanced environments closer, we realize that their overall session setup, establishment and run-time session management are very complicated, labor intensive, and incomplete since they are still based on older session initiation and management protocols and standards for single streams such as IETF SIP, RTP/RTCP, RTSP, SDP protocols, which do not take into account efficiently new requirements and demands coming from 3DTI environments. The new 3DTI requirements are as follows: We are dealing with cyber-physical rooms encompassing *large scale* of multi-sensory devices that generate large scale of real-time streams across geographically-distributed locations, and interact with each other in synchronous and real-time manner. It means that the 3DTI rooms generate *highly correlated* multi-sensory content that *requires* to be captured in *synchronous* and *real-time* manner, as well as forwarded in such a way that it gets rendered and presented at output device(s) in synchronous and real-time manner.

To deal with (1) large scale of correlated sensors, (2) real-time and synchronous capture, transmission and rendering, and (3) interactivity and flexibility of 3D immersive data, we need a new and advanced session management with (a) *session initiation protocol(s)* that understand media co-dependency and correlations since a logical session connection is represented by a bundle of streams, not a single stream, (b) *multi-stream topology establishment* among multiple parties since different users might request different sensory streams (i.e., different view) from other sites, not the same video is delivered to everybody in the session, (c) *efficient, light-weight and distributed session monitoring* with querying and debugging capabilities since we need online investigation and debugging when some of the sensors fail during the session run-time, (d) *flexible session adaptation with efficient topology adjustments* since users do not get all sensory information all the time and may change views including different sensory information, and (e) *light-weighted and consistent session tear-down protocols* since different parties may leave at different times.

In this paper we discuss a new session management and new session services for 3D Teleimmersive Interactive (3DTI) environments that are based on our six years experiences with the TEEVE (Teleimmersion for Everybody) system [38]. We will present the needs for a *registration service* of sensors (I/O devices such as array of cameras, arrays of displays) during the session initiation, *session establishment service* of view-based stream topologies

that will give the freedom to users to get desired view information, session monitoring, querying and debugging services that will make it much easier to acquire run-time situational awareness of each media-rich session, and *view-casting service* for session adaptation to allow users to change their media views during the 3DTI run-time session. The presented design of the next generation session management protocols, services, algorithms and data structures is based on our extensive experiences with building 3D teleimmersive interactive systems such as TEEVE [35, 38, 42], experimenting with high impact teleimmersive applications [3, 25, 37, 39] and deploying such environments at various venues [31]. Hence, we are confident that it gives a strong base for future session management frameworks, their designs and standardization efforts that must take place in order to simplify the setup and management of these up-coming media-rich interactive environments.

The discussion of the next generation session management is divided as follows: In Section 2, we present the 3DTI model, its system architecture, 3D immersive data and view models, and the session metadata that each 3DTI application and underlying service middleware component yield. Session management overview and road-map with the individual functions is provided in Section 3. Using the road-map of Section 3, we dive into detailed discussions on session initiation protocols, algorithms and services in Section 4, on session monitoring algorithms, protocols and services in Section 5, and on session adaptation with view-casting management in Section 6. Section 7 completes the detailed discussion about the session management with description of the tear-down protocol and session termination. Section 8 puts our presented session management framework into perspective with other session management frameworks that are currently available. We conclude with Section 9, calling for future research attention and standardization in this space.

2 3D teleimmersion model

Our 3DTI environments will be modeled according to the TEEVE system built and deployed at the University of Illinois, Urbana-Champaign. The TEEVE (Teleimmersive Environment for Everybody) system [38] bears several important and representative features including multi-stream 3D capturing and transmitting, multi-party collaboration, overlay network and QoS management. It has also been constantly used in large number of real applications and user studies over the years [3, 25, 37, 39]. That is why we choose TEEVE as the reference system.

The 3DTI system is built on a layer of distributed service middleware which contains multiple tiers. The *capturing tier* consists of camera units, microphones, body sensors and outputs synchronized and correlated 3D video streams and other media streams at one site (sender-site). The *transmission tier* is responsible for transmitting streams over the Internet among geographically distributed sites. The *rendering tier* receives streams from different sites and renders them as one single virtual environment using synchronized multiple displays, speakers and haptic devices at one site (receiver-site). At each tier, the session management deploys algorithms, services and protocols that rely on clear data model, system model and service architecture which is described in this section. As we discuss further models, session management algorithms, protocols and services, we will refer only to arrays of cameras on the input side and multiple displays on the output side, but it is important to stress that the models can be extended to other sensory devices and media streams.

2.1 Data model

A teleimmersive environment installs multiple 3D camera units and has basic hardware for synchronizing them. Accordingly, the data model consists of two parts: (a) 3D reconstructed video data model that represents the information coming out from one camera, and (b) integrated data model that includes all captured 3D contents at one site from multiple cameras at the same time instant.

As shown in Fig. 1, a 3D camera unit is a cluster of 2D digital cameras. Multiple 3D cameras are mainly mounted on two levels of metal frames, with the top ones (about 6 ft) capturing the upper body, and the lower ones (about 3 ft) capturing the lower body. Each 3D camera is hosted by a computer which handles image grabbing and 3D reconstruction. The data captured from a camera cluster at a particular time instant t represents one 3D frame of the scene, denoted as $f_{i,j}^t$, where i is the identifier of the teleimmersive environment and j the identifier of the camera. A 3D frame contains both color and depth information for every pixel. The depth is derived by the trinocular stereo reconstruction algorithm using images captured from synchronized black/white cameras [1]. We define a 3D stream, $s_{i,j}$, as a temporal sequence of 3D frames (i.e., $s_{i,j} = \{f_{i,j}^{t_1}, f_{i,j}^{t_2}, \ldots\}$).

The integrated data model relies on both spatial calibration and temporal synchronization of cameras such that 3D frames from different cameras will form a consistent and comprehensive view of the scene. One approach is to use the multi-camera self-calibration toolkit for both intrinsic and extrinsic calibration of all cameras [28]. A trigger server is used to synchronize all cameras to grab images at the same instants of time. This is done by periodically sending a hardware trigger signal from the parallel port on the server to a pair of general purpose I/O pins on each camera. The integrated data model indicates that at each time instant the capturing tier must have multiple 3D frames representing a wide field of view. The frames bearing the same timestamp constitute one *macro-frame*, denoted as F_i^t, where $F_i^t = \{f_{i,1}^t, f_{i,2}^t, \ldots\}$. Thus, the teleimmersive system yields a stream of macro-frames at certain frame rate.

Multiple 2D and 3D displays are placed in different positions and angles to present the scene such that the teleimmersive participant can have more observation freedom when he/she moves or turns (Fig. 1). Each display is connected to a computer which handles real-time rendering of the 3D video streams. It also provides a user interface to change the rendering viewpoint in the virtual space like rotation and translation. This is important

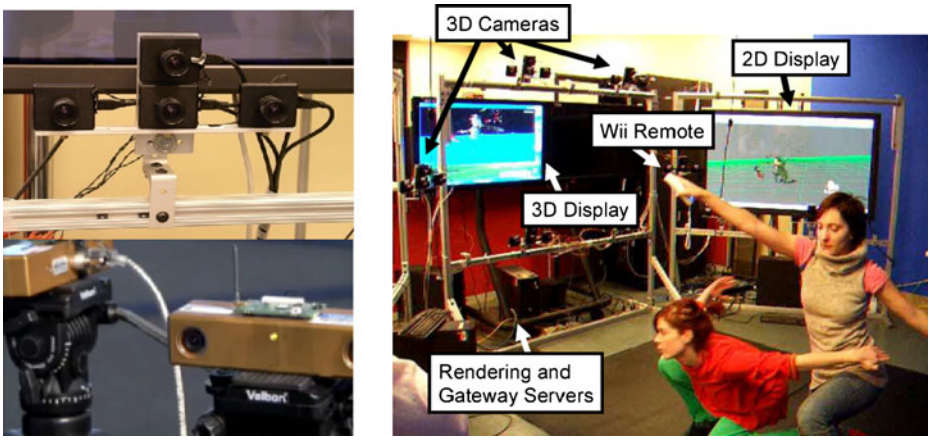

Fig. 1 3D cameras (old and new versions) and Illinois teleimmersive environment

because the power of 3D data representation lies in its capability of allowing users to watch the virtual space from arbitrary view angles.

2.2 View model

One unique characteristic of multi-stream 3D video content lies in *stream correlation* due to the fact that all 3D cameras are concurrently capturing visual data with a synchronized clock and presenting complementary visual information of a common physical scene. However, since a user only observes the visual information from one particular view at any given time and each 3D stream only conveys part of the whole content from its own viewpoint, the *contribution* of each stream to the current user view must be different. Such view-dependent stream differentiation can be formally represented using the view model.

We are interested in deriving an appropriate indicator of the stream importance regarding to the user view. One example of the stream differentiation function is given in [40]. Let us denote $s.\vec{w}$ as the unit vector representing the spatial orientation of a 3D camera and its corresponding stream s, and $u.\vec{w}$ as the unit vector representing the view angle of user u. The stream differentiation function, denoted as df, calculates the stream importance as in Eq. 1.

$$df(s,u) = s.\vec{w} \cdot u.\vec{w} \tag{1}$$

In the equation, the value of the dot product equals to $cos\theta$, where θ is the angle between the vectors of $s.\vec{w}$ and $u.\vec{w}$. When $s.\vec{w}$ and $u.\vec{w}$ are close to each other, the dot product is close to 1, showing that stream s is very important to the user view. The value decreases to -1 indicating less importance, and vice versa. As shown in [40], the quality adaptation can be achieved by simply dropping the streams whose importance is below a certain threshold.

2.3 System model

The teleimmersive system model has three layers. The top layer is the *application layer* and it is responsible for the management of end-devices including cameras, displays and interfaces for 3D reconstruction, rendering and user interaction. The *distributed service middleware layer* represents the more important component of the system. The bottom layer of the 3DTI system is the *networking layer* build on end-system overlay concepts.

Figure 2 illustrates the architecture of the distributed service middleware from the aspect of teleimmersive session. The architecture features a hierarchical design of two basic components: *service gateway* and *session controller* for the scalability of supporting real-time physical collaborative activities among geographically distributed environments. Each

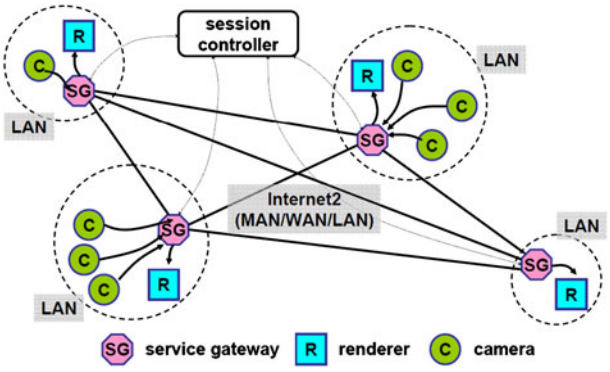

Fig. 2 Architecture of distributed service middleware

environment is managed by its local service gateway. Globally, service gateways are managed by the session controller which handles the membership and overlay topology construction. The service gateway collects multiple streams either from the local sources (e.g., cameras) or from the remote sources (i.e., peer service gateways). The collected streams are then multicasted according to the current status of the overlay network, the view change requests and the forwarding schedule coordinated by the session controller. Together, service gateways and session controller perform important session management functions including session initiation, monitoring, adaptation and termination. We take a centralized approach to the session management at the global level because of its low messaging cost and responsiveness to the dynamics of a running 3D teleimmersive session. The approach is feasible in our situation since the number of 3DTI sites participating in an interactive session is not expected to exceed 10 (i.e., the number of service is expected to be within a reasonable scale (≤10)).

2.4 QoS metadata model

Various components connected to the environments generate QoS metadata during the operational (run-time) phase of the teleimmersive session as shown in Fig. 3. We classify the QoS metadata into *system* and *application* metadata. Depending on the metadata update frequency, we classify them into *dynamic* and *static* metadata. Static metadata are configured and fixed from the beginning of the teleimmersive session and dynamic metadata changes frequently while the session is running. Both dynamic and static metadata influence the QoS adaption.

Metadata information about each device, resource or environment are expressed as a *data item*. We define a data item d as a set of (*attribute*, *value*) pairs and service gateway description, servicing the data item d. It means, $d = \{(a_1, v_1), (a_2, v_2), \ldots, (a_n, v_n), I\}$, where v_i gives the value of attribute a_i and I is a tuple, *itemInfo*, describing the service gateway. For example, the information of a camera at the gateway "teeve1.cs.xyz.edu" can be expressed using {(framerate, 10), (shutter, 120), (gain, 133), (zoom-level, 12), (3D-reconstruction time, 10), (avg. memory usage, 78%), (avg. CPU utilization, 34%), itemInfo}, where itemInfo=(node-info=camera-node, camera-id=2, host-gateway=teeve1.cs.xyz.edu, OS=RHL, Kernel version=2.2.6). Here, itemInfo is not an attribute about the device but rather provides environmental information about the service gateway where the device belongs to.

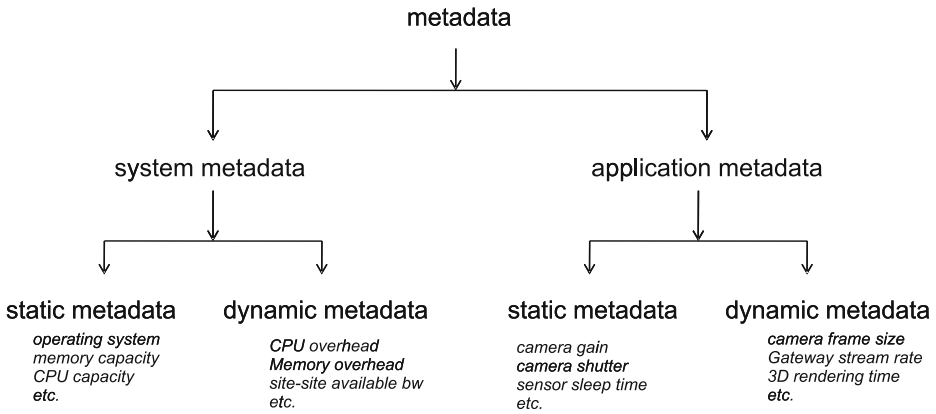

Fig. 3 Teleimmersive system metadata classification

3 Session management overview

The session management is implemented as a suite of distributed protocols to address the complexity of connecting multiple 3D teleimmersive environments. This section will provide a roadmap for the session management design and its protocols. As mentioned in Section 2, all of these protocols are located in the distributed service middleware layer of the 3DTI system and executed jointly by service gateways and the session controller.

The session management protocols can be divided into two groups: (a) those assisting 3D video content dissemination, and (b) those executing the overall system monitoring via metadata measurements. The session management protocols are initiated during the session setup stage to fulfill their future functions in the data and control plane, respectively. The 3D data dissemination is critical for the 3DTI system involving the delivery of 3D content-data and meta-data in real time. The system monitoring is responsible for collecting metadata related to system performance and its health. At this point, the 3D data dissemination is assisted by session management protocols including *session initiation*, *session adaptation* and *session termination*, while system monitoring contains the protocol of *session monitoring*. The following sub-sections in Section 3 will give an introduction to each protocol including major functions and algorithms. More detailed description of each protocol will be provided in individual Sections 3, 4, 5, 6, 7.

3.1 Message format

The session management specifies a general message format as the following. (Fig. 4)

The *type* field indicates the message type. The *length* field gives the size of the *payload* field, whose content depends on the specific message type. The implementation of distributed session protocol relies on the exchange of the messages.

3.2 Session initiation

The session initiation protocol deals with the establishment of a teleimmersive session at two levels. Within one environment, the service gateway keeps track of its local resources and serves as an aggregation point for both data and control flows. For outbound traffic, the service gateway collects local streams from the camera host PCs, and disseminates them out to other service gateways across the network. For inbound traffic, it receives streams from remote environments, and distributes both local and remote streams to the local displays.

At the bootstrapping stage, the service gateway is launched first. The end-devices such as camera host PCs and display host PCs register with the service gateway using the *join* message. The type field (described in Section 3.1) of the join message could be CAMERA_JOIN or DISPLAY_JOIN accordingly. The payload field holds the meta-data of the camera and display. For camera, it contains the intrinsic and extrinsic parameters, whereas for display it contains the user view information used for view-based content dissemination and adaptation (more details later). The service gateway maintains a resource table with each entry representing one site's end-devices. After the local environment is established, the service gateway connects with the session controller using a similar join

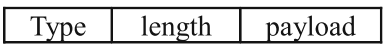

Fig. 4 Message format of the session protocol

message with the type field set as GATEWAY_JOIN. The payload of the message contains a summary of local 3D streams and user views.

Another important task in the session initiation is to set up a simple yet efficient content dissemination topology based on the overlay network between service gateways. The initial topology construction leverages the distribution of user interests to efficiently utilize limited network resources. Due to the complexity of the problem, several heuristic topology construction algorithms have been proposed using, for example, the largest tree first and randomized policies [35]. More details are presented in Section 4.

3.3 Session monitoring

With the increase in system scale in terms of number of devices and components connected at each environment, it becomes really hard to manage and monitor the whole 3D teleimmersive system from a single administrative point. Queries in such systems are not like the traditional database queries with a single key value, instead they are given in a high level description which are transformed into multi-attribute composite range queries such as "which environment is highly congested?". To answer that, the query is transformed into a multi-attribute composite range query with constrains (e.g., range of values) on CPU utilization, memory overhead, stream rate, bandwidth utilization, delay and packet loss rate. Another mentionable property of such systems is that data traffic is real-time high bandwidth multimedia traffic and consumes significant part of the communication bandwidth. So, the session monitoring system should be light-weight, scalable in terms of data items and capable of answering queries in low latency.

We present Q-Tree, the multi-attribute range-based query solution considering above mentioned requirements [2]. One of the significant properties of Q-Tree is that it injects only a single query to the overlay for any size of composite multi-attribute queries without any preprocessing and still ensures the optimal number of node traversal. It can handle significant amount of attribute churn in the teleimmersive system and also scales with the number of data items. Two types of data attributes are usually found: *static* and *dynamic*. Static attributes (such as static camera parameters) do remain fairly unchanged over an entire teleimmersive session, whereas dynamic data attributes (such as frame rate, end-to-end delay, CPU utilization, bandwidth etc.) change frequently. Any arbitrary node may initiate a query either providing a high level description of the query or explicitly defining any combination of (attribute, value/range) pair. The underlying requirement of serving queries is to retrieve data items from relevant nodes. Q-Tree organizes service gateways in an overlay tree structure and assigns ranges to nodes in some hierarchy. Data items are disseminated into the overlay to be stored remotely in some other nodes according to the value. When a query is made for items specifying the range for certain attributes from any arbitrary node, a distributed search is initiated across the overlay. The primary objective of the query is to locate those nodes that store the requested data items. The tree structure with hierarchical ranges makes this query to be served efficiently. Data items with dynamic attributes higher than the bounded rate of update are handled via multicast. More details of session monitoring are presented in Section 5.

3.4 Session adaptation

The session adaptation protocol dynamically maintains the content dissemination topology from its initial construction. The protocol function is carried out mainly by the session controller using a view-based approach called ViewCast [42]. The view concept reflects the

user interest at a higher level. When a user retrieves 3D content from the system, only the user's view interest is required as the searching criterion. The ViewCast scheme controls the stream selection dynamically according to the view requirement and the status of resources with the ultimate goal of sustaining the QoS for the rendered view. Under the scheme of ViewCast, the network resources needed for streaming are allocated according to the importance of streams differentiated by the view model (Section 2.2). As the user view, content distribution and network state keep changing, the session adaptation protocol responds by modifying the content dissemination topology.

One critical challenge of ViewCast is how to deal with *view change* efficiently. As observed, the view change operation may occur frequently in 3DTI environments. There are two consequences associated with the view change. First, the stream differentiation varies with view change (i.e., function *df* not fixed). The feature distinguishes ViewCast from other systems that only consider the fixed stream differentiation such as the layered coding and the multiple description coding. Second, as soon as the view change is detected, the system needs to respond by reallocating streaming resources accordingly. This reallocation procedure could be costly under certain circumstances. The direct impact to the user may be the discontinuity or delay of view rendering. If the multicast protocol is used in the underlying layer, then the resource reallocation at the parent node may influence child nodes. The challenge is addressed by several techniques which aim to relieve the severe impact of view change.

The ViewCast protocol involves several messages including VIEWCAST_REQUEST, VIEWCAST_REPLY, STREAM_REQUEST, and STREAM_REPLY. The content of payload field depends on each message type. For example, it may contain user view information as in VIEWCAST_REQUEST or the stream information as in STREAM_REPLY. Those messages are exchanged between display host and service gateway, service gateway and session controller, and service gateways themselves. More details are given in Section 6.

3.5 Session termination

The protocol of session termination provides the mechanism for any component including camera, display and service gateway to leave the session voluntarily. The message types include CAMERA_LEAVE, DISPLAY_LEAVE and GATEWAY_LEAVE. Note, the leave of one component may trigger messaging in other protocols. For example, when a display host leaves a session (i.e., DISPLAY_LEAVE), it may indicate that the contents for a particular user view are no longer needed. In such case, another VIEWCAST_REQUEST message will be submitted to the session controller to update the content dissemination topology if necessary. More details are presented in Section 7.

Note that, the involuntary leave of teleimmersive components is handled by the session monitoring protocol based on periodic heartbeat messaging and timeout mechanism. Both service gateways and the session controller maintain soft-state information which can be erased once failure is detected.

4 Session initiation

As mentioned earlier, the session initiation protocol for 3DTI deals with two major tasks: (a) the registration of various devices and resources and (b) the construction of an initial content dissemination topology based on the overlay network formed between service gateways. These two tasks are introduced in this section.

4.1 Session establishment

The 3D teleimmersion session starts at each environment. The service gateway is launched first. The camera host PCs are then connected and registered with the local service gateway using the join message of CAMERA_JOIN. The payload field of the message contains the meta-data of the camera (e.g., the intrinsic and extrinsic parameters). The service gateway maintains a *stream table* to store the information for each camera. In principle, each 3D camera unit is a separate component which can join (or leave) the session independently. However, to simplify the design of the hardware synchronization of cameras, it is required that all cameras intended for one session join at one single stage. The number of cameras to be used is a configuration parameter given to the service gateway when it is launched.

The service gateway waits until all 3D camera units have joined. After that, the cameras start to transmit reconstructed 3D video streams to the gateway via high-speed LAN and the gateway becomes a teleimmersive content source. At this point, if a display PC joins the session after registering with the gateway (using the DISPLAY_JOIN message), it will receive 3D video content from the gateway (although only one environment).

From the view of service gateway, each display device represents one user *view* into the teleimmersive virtual space. The DISPLAY_JOIN message contains the view information so that the gateway can perform view-based stream selection and content adaptation (more details later). The DISPLAY_JOIN message also includes other information describing the rendering capacity of display. For example, the service gateway can accommodate mobile devices of limited graphics processing power by transmitting rendered 2D streams to them instead of 3D streams [26]. Similar to cameras, the service gateway maintains a *display table* to maintain the information for all local displays.

To join a multi-environment session, the service gateway needs to register with the session controller using the GATEWAY_JOIN message. The join message contains the meta-data of local 3D streams. The session controller is responsible for propagating the information among all currently active service gateways. For this, the session controller maintains a *gateway table* to keep track of active service gateways and their information.

It is necessary for service gateways to differentiate local versus foreign streams which are handled differently. A user view request for local streams is processed by the service gateway directly. Such request never goes to the session controller. However, a view request involving the transmission of foreign streams (i.e., streams hosted in remote gateways) must be forwarded to the session controller where algorithms are executed to derive an appropriate dissemination topology based on the current networking status and content distribution. For example, the session controller may ask an intermediate gateway to relay streams instead of requesting the original source.

4.2 Construction of initial content dissemination topology

The major concern for the initial 3D content dissemination topology is the efficient or optimal construction of the *3D data transmission overlay topology*, subject to multiple constraints including bandwidth and latency. The topology construction needs to be carefully managed because the resources are shared among all environments. The construction of 3D data dissemination topology here assumes an initial set of user view requests. It can be used in situations where user views are rather fixed for each session. For example, the user view of previous session can be recorded and used in the next session. On the other hand, the management of dynamic user view changes is handled by the session adaptation protocol (more details in Section 6).

As shown in Fig. 5a, within each environment the service gateway forms a star topology to the cameras and displays. However, the construction of the overlay among all gateways on the WAN poses a key challenge. The main goal is to organize an overlay We define the multicast group, $M(s)$, as the set of gateway nodes that have requested (i.e., one of its displays) the stream s. In such multi-stream/multi-environment scenario, the overlay topology, to be constructed, is essentially a forest of multiple trees, with each tree designated to disseminate a stream among the set of requesting gateways. More specifically, for each multicast group $M(s)$ of stream s, a multicast tree T_s, needs to be constructed to disseminate the stream from the source to all other nodes. Figure 5b shows an overlay topology of four environments with four trees constructed among the gateways of $\{A,B,C,D\}$. The label on each edge denotes the index of the source gateway for the transmitted stream. For instance, gateways *A, B, C* and *D* form a multicast group $M(s_B)$ that transmits the stream originated from gateway *B*, while gateways *A*, *B*, and *D* form another group $M(s_A)$ for stream from *A*. Note that this is only a simplified example, as each node is the source of a single tree. In reality, each site often has tens of streams to disseminate, and hence it acts as a source for that many trees.

The construction of such a data transmission overlay forest is complicated by several characteristics of 3DTI environments: (a) multiple system constraints: each node has inbound and outbound bandwidth limits, and the end-to-end delay between any pair of nodes has to be small in order to guarantee interactivity; (b) a dense graph: since the participant typically wants to see a large portion of other participants from a wide field of view, the overlay graph consisting of all environments often has very high density (i.e., the average in/out-degrees of all nodes are large); hence, the construction of the forest needs to be carefully coordinated.

4.2.1 Problem formulation

Due to the huge demands of computing and networking resources in multi-site 3DTI collaboration, we have two constraints and one optimization goal to satisfy in the initial overlay construction problem.

Constraint I (bandwidth) Each node v has inbound (I_v) and outbound (O_v) bandwidth limits in the unit of number of streams (i.e., $I_v, O_v \in N$), which can be dynamically measured by existing probing tools like Pathload [13]. A service gateway should never receive more data

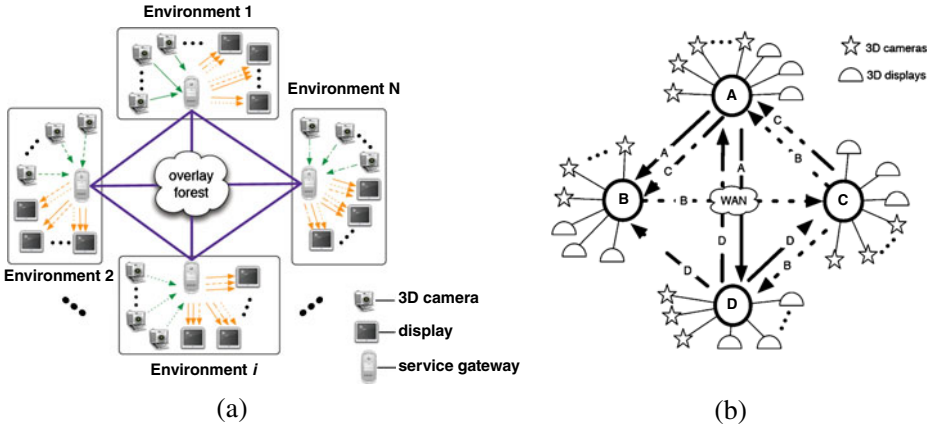

Fig. 5 **a** Multi-stream/Multi-environment 3DTI, **b** Content Dissemination Overlay Topology

than its inbound bandwidth limit (i.e., $d_{in}(v) \leq I_v$, where $d_{in}(v)$ is the actual in-degree of node v in the overlay), nor be delegated to send data more than its outbound bandwidth constraint (i.e., $d_{out}(v) \leq O_v$, where $d_{out}(v)$ is the actual out-degree of node v).

Costraint II (latency) In 3D teleimmersive session, remote participants are rendered into the cyber-space in real time for interactive collaboration. Therefore, the expected end-to-end latency or cost between any pair of nodes v and u, $cost(v,u)$, should not exceed certain bound, B_{cost}, in order to guarantee interactivity.

Optimization goal (request rejection ratio) Due to the two stringent constraints listed above, we cannot guarantee that all subscription requests are satisfied. The metric we wish to minimize is the total rejection ratio of all requests in the system, denoted by X. Suppose the number of requests made by node v to u is $r_{v \to u}$, among which $\widetilde{r}_{v \to u}$ are rejected, we thus define the rejection ratio as in Eq. 2.

$$X = \sum\nolimits_{v} \sum\nolimits_{u, v \neq u} \frac{\widetilde{r}_{v \to u}}{r_{v \to u}} \tag{2}$$

More specifically, the *forest construction problem* can be formulated as follows.

Forest construction problem Given (a) a completely connected graph $G=(V,E)$, (b) an in-degree bound $I_v \in N$, and an out-degree bound $O_v \in N$, for each node $v \in V$, (c) $\text{cost}(u,v) \in Z^+$ for each edge $e \in E$, which denotes the latency, and (d) a set of multicast groups $M = \{M_i | M_i \subseteq V\}$, each with a source $S(M_i) \in V$, the goal is to find a *spanning forest*, $F = \{T_i | T_i \subseteq G\}$, with each tree T_i being a spanning tree that spans from the source $S(M_i)$ to a subset of the other nodes (i.e., $M_i^{'} = M_i - S(M_i)$), such that the total fraction of excluded nodes, $\sum_i \frac{|M_i - M_i^{'}|}{|M_i|}$, is minimized, subject to the constraint that for all $v \in T_i$, $d_{in}(v) \leq I_v$ and $d_{out}(v) \leq O_v$, and $cost(S(M_i), v)_{T_i} \leq B_{cost}$.

Wang et al. [33] proved that the problem of finding a solution subject to two or more constraints in any combination in the multicast routing problem is NP-hard. In light of this, we study several heuristic algorithms to address the problem.

4.2.2 Stream selection and differentiation

Before the creation of the initial spanning forest, we need to map the set of user requests into a set of *stream requests*. The mapping takes two basic steps: *stream selection* and *stream differentiation*.

Stream selection Given a user view request the importance of streams is calculated using the differentiation function (i.e., *df*) based on the view model (Section 2.2). The streams are selected if the *df* value is above a certain threshold. For example, we can choose the streams with $df(s,u) \geq 0$, accommodating a 180° total viewing range.

Stream differentiation The selected streams are further differentiated into several priority groups according to their importance. We define the set of m priorities P as $\{p_1, p_2 ..., p_m\}$, where $p_i < p_{i+1}$ for $1 \leq i \leq m-1$. We assign priorities to the selected streams according to the differentiation function. That is, we sort the selected streams according to the value of the differentiation function. The stream with the largest value of the differentiation function is assigned the highest priority p_m, the stream with the second largest value is assigned the

priority p_{m-1}, and so forth. The result is that each priority group typically contains one stream from each environment.

4.2.3 Heuristic algorithms

After the stream requests are selected and differentiated, the tree-based algorithm then works on them to create the initial content dissemination topology. Several tree-based algorithms can be applied which differ in the order in which they construct trees, namely, *largest tree first* (LTF), *smallest tree first* (LTF), *minimum capacity tree first* (MCTF), and *randomized join* (RJ) [35].

LTF The intuition is to construct the largest tree first so that even if the last few trees cannot be constructed due to saturation, the rejection ratio should be small because we are left with the smallest trees. The size of a tree T_s is intuitively the number of nodes in the corresponding multicast group (i.e., $|M(s)|$). Specifically, we first sort all multicast groups based on the size, and then construct the spanning trees one by one from the largest multicast group to the smallest one.

STF As a comparison to LTF, this algorithm starts from the smallest multicast group, and ends with the largest one. The hypothesis is that the rejection ratio of LTF should be smaller than that of STF.

MCTF This algorithm considers the difficulty of tree construction in terms of the forwarding capacity of a tree. The intuition is that the larger this value is, the easier it is to construct the tree. That is because new requests are easier to accommodate with a tree containing large aggregate forwarding capacity (i.e., out-bound bandwidth). The forwarding capacity of a tree, T_s, is the sum of the forwarding capacity of all nodes in the multicast group $M(s)$. This algorithm sorts all multicast groups in the ascending order based on the aggregate forwarding capacity, and starts from the multicast group with the least capacity, to the one with the largest.

RJ LTF, STF, and MCTF all seek to build the trees one by one, that is, only when it finishes processing all requests in one tree will it move on to construct the next one. In contrast, we propose a randomized algorithm which randomizes all requests for the whole forest, with no prioritization on any tree.

The experimental results in [35] find that RJ generally outperforms the other tree-based algorithms. One reason that the randomized algorithm works better is that every node in a 3DTI session is likely to be overloaded with user view requests, because a participant typically wants to see a large portion of other participants from a wide field of view. In tree-based algorithms, a node is much more likely to be congested in the first few constructed trees if it is the source, or a node near the source. This increases the probability of rejection in the construction of the latter trees because the node's total bandwidth is shared among different trees. In contrast, the randomized algorithm achieves good load balancing because it distributes the tasks of request processing among different trees randomly.

5 Session monitoring

The session monitoring protocol is used to keep track of the performance metadata and maintain the system fault tolerance in terms of component failures during the 3D data

dissemination and overall session run-time. To accomplish that in a distributed fashion, the protocol includes the design of monitoring overlay management, query formulation and processing, churn and failure handing, and load balance.

5.1 Building management overlay

The session monitoring/management overlay for metadata dissemination should consider efficient paths to make less interference with the overlay forest for multi-dimensional teleimmersive 3D data streams. We present the *metadata overlay topology construction* considering the round-trip time (RTT) between the nodes. It is assumed that the session controller knows RTT delays among all pairs of nodes. This can be maintained by a couple of ways. For example, an individual node may periodically report its measured latencies to a set of other nodes or derive its network coordinate [7]. Given a complete graph with edge costs, the session controller computes the optimal management (monitoring) overlay structure for the entire system and then disseminates this information to all nodes. We consider several options for optimal management overlay construction.

Minimum spanning tree (MST) is considered as one of the best options for multicast overlay. One potential problem with MST could be its load imbalance that it may eventually end up of becoming a star, and a single node becomes a point of contact for most queries. Degree bounded MST (DBMST) requires that no node has degree more than some constant k which relieves the load of a single node. However, DBMST is NP-hard for any given k [10]. We present one solution which is a modification of the classical Prim's algorithm to achieve an approximation for DBMST. Prim greedily selects the lowest cost edges expanding a small seed tree. The modified Prim includes an edge only if it does not violate the degree bound. To prevent the generation of long chain of nodes, we restrict the length of each chain and build a height balanced k-array tree. An example of overlay construction with five environments is shown in Fig. 6.

As described in the session initiation, when a new environment joins an on-going teleimmersive session the service gateway of the new environment sends a join message to the session controller. The session controller handles the registration and attaches the new gateway (say 3) to an existing node (say 1) into the session monitoring overlay considering its locality and degree bound of the existing node. (i.e., node 1 is the closest node of node 3 in the existing overlay that has less than k-number of neighbors). Node 1 becomes the parent of node 3.

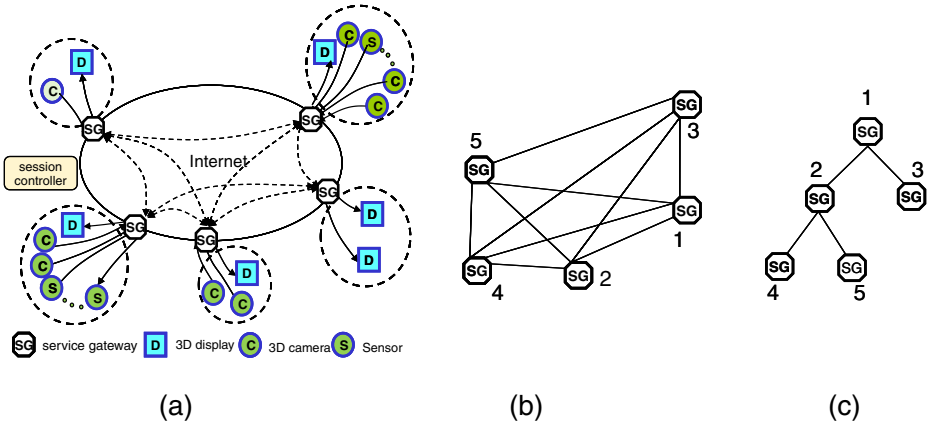

Fig. 6 **a** 5 3DTI environments, **b** Fully connected gateway overlay (the number signifies the geographic distance of nodes), and **c** Constructed management overlay

5.2 Range assignment for QoS metadata distribution

After the management overlay tree has been constructed, the session controller assigns range intervals to each service gateway to distribute metadata and speed up the query resolution. These ranges are distributed among nodes along the tree hierarchy, and they are mutually exclusive.

A range r is denoted as $r(l,h]$, where l and h are the lower and higher limit, and $0 \le l < h \le 1$. The value, v, lies in r (i.e., $v \in r$) if and only if $l(r) < v \le h(r)$. A range r_1 is contained in r_2 (denoted as $r_1 \prec r_2$) if $l(r_2) \le l(r_1) \wedge h(r_1) \le h(r_2)$. Two ranges r_1 and r_2 are said to be consecutive if $h(r_1) = l(r_2) \vee l(r_1) = h(r_2)$. Two consecutive ranges can be unioned as: $r = r_1 \cup r_2$ where $r = (\min(l(r_1), l(r_2)), \max(h(r_1), h(r_2))]$. We denote $|r| = h(r) - l(r)$ as the length of the range.

Let T be the rooted tree, where $P(x)$ and $C(x)$ are the parent node and the set of child nodes of node x. Each service gateway (SG) node x is assigned with two ranges, self-range $\delta^a(x)$ and subtree-range $\Delta^a(x)$ for each metadata a. Subtree-range $\Delta^a(x)$ specifies the range assigned to the entire subtree rooted at node x. An item d with an attribute-value pair (a,v) is stored at node x if $v \in \delta^a(x)$. If $v \in \Delta^a(x)$, then d is stored somewhere in the subtree rooted at node x. By definition, $\delta^a(x) \prec \Delta^a(x)$. The following four properties are held by the ranges assigned to nodes for any distributable metadata a:

- Disjointness: $\delta^a(x) \cap \delta^a(y) = \emptyset$, for any pair of nodes x and y.
- Subtree range: $\delta^a(x) \prec \Delta^a(x)$.
- Hierarchy: If x is an ancestor of y, $\Delta^a(y) \prec \Delta^a(x)$.
- Entirety: $\Delta^a(root(T)) = (0.0, 1.0]$, where $root(T)$ is the root node of T.

Algorithm 5.1 shows how ranges are assigned to nodes. The assignment is initiated by the root by invoking $AssignRange_{root}$(a, 0.0, 1.0). Individual node assigns range to itself and to nodes in its subtrees, similar to a preorder traversal of the tree.

Algorithm 5.1 $AssignRange_x(a, l, h)$

Input:
  x: node to which assignment is being made;
  a: attribute;
  l, h: real numbers, range bound;
  N: number of nodes

begin
  $\Delta^a(x) \leftarrow (l, h]$;
  $\delta^a(x) \leftarrow \left(l, l + \frac{1}{N}\right]$;
  $l \leftarrow l + \frac{1}{N}$;
  for all c **in** $C(x)$ **do**
    $n_c \leftarrow c.size$; /* size of the subtree at child c */
    $h \leftarrow l + \frac{n_c}{N}$;
    $AssignRange_c(a, l, h)$;
    $l \leftarrow h$
  end for
end

We assumed that all values within the entire range (0.0,1.0] of an attribute are equally likely. So, each node gets a self-range of equal size $|\delta^a(x)|=1/N$ for each attribute. This ensures that each node stores nearly the same amount of data items. But if it happens that certain attribute follows some distribution other than uniform, the load may get skewed and in that case the load balancing is invoked to change the range of the nodes accordingly. The *query engine* at each service gateway stores the local range, parent range and the children ranges.

5.3 Metadata normalization and distribution

We assume that all metadata values can be normalized to (0.0, 1.0]. This can be easily done if the domain experts can specify the minimum and maximum possible values for each attribute. Then, a simple calculation of $(v - v_{\min})/(v_{\max} - v_{\min})$ normalizes the value v to (0.0,1.0]. Non-numeric metadata (such as OS specifications) can also be mapped to some integer value (e.g., hashing) and normalized using the above equation. Such metadata are distributed along with the local environment and node information into the overlay.

While the session is running, each activity in the system generates corresponding metadata. This metadata is fused into the range tree considering its range along with the *itemInfo*. *itemInfo* field is necessary. When a query finds a metadata at a remote node, it contacts to the source gateway of this metadata to justify the validity. This is because when a change happen to this metadata (e.g., a new sample), the update instantaneously goes into the overlay without deleting the previous entry. Each metadata uses a periodic *refresh interval* to check its validity by contacting the source node. If the metadata is no longer valid, it is removed from the management overlay. Any time, a query is served by any node (service gateway); the refresh-interval of the metadata inside the query gets updated. Fig. 7a shows an example of inserting distributable metadata into the overlay structure to disseminate the data according to all the view requests.

5.4 Metadata query

Once data items are fused, the management overlay is ready to serve queries. Queries could be of different types. The first type is *instance of time* query that asks for the value of different system parameter at the running instance of time. For example “How many cameras are on at Site 2?”, “What is the current view plan of the users at Site 5?”, and “What is the end-to-end bandwidth between service gateways A and B?”. The management

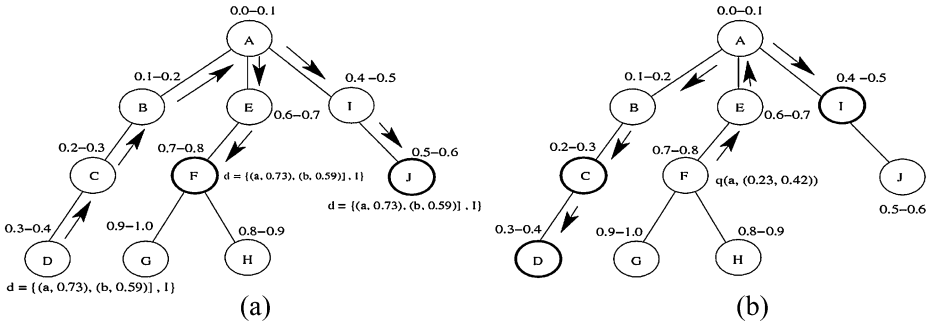

Fig. 7 **a** Insertion of data item *d= {(a, 0.73), (b, 0.59), I}* originated at node *D*. **b** Query *q* of metadata attribute *a,* specifying the normalized range (0.23, 0.42), and originated at node *F*

overlay system also supports *aggregation* query such as COUNT, MAX, MIN, and AVG, satisfying query such as "What is the average frame size of the system?". An interface exists to monitor *continuous probing* query such as "Monitor the frame rate dynamism at Camera 2".

Queries for distributable metadata are routed considering the ranges assigned to the gateways into the management overlay. The overlay is especially designed for range queries with single or multiple attributes. Any node can originate a query by initiating a distributed search to locate nodes where the requested data items are stored. In all cases, replied items are returned to the query source being aggregated in the intermediate nodes along the path of the tree. An example is given in Fig. 7b showing how the query is propagated into the management overlay for distributable metadata.

To serve the query for non-distributable metadata, the query goes to the specific node (if the destination is given in the query) via tree overlay and it returns the result on the same path. If the destination node information is not given, the management overlay system multicasts the query and returns the reply back via tree aggregation. The notable feature in this management overlay is that a single tree stores multiple different metadata and to query a complex combination of metadata, only a single query needs to be fused into the overlay. Details of the query algorithms within the management overlay are given in [2].

5.5 Failure and churn handler

For failure detection, each node/gateway keeps a *nodebuddy* list and sends periodic heartbeat messages to monitor all the nodes on the list. Children of a node are the nodebuddies of the parent, and the root is monitored by one of its children. Every failure or departure is reported to the session controller which then removes that node from the overlay and replaces it by one of the child nodes. This procedure may need the modification in the range value. While replacing a failed node, we select the child that has consecutive range to the failed node. The new node parent will have the sum of inclusion of both ranges. In case of voluntary departure, the parent transfers all distributable metadata that it was storing as part of the range before it leaves to the potential child. Otherwise, the new parent makes a range query of the range of failed node to get the failed distributable metadata restored.

5.6 Load balancing

As highly dynamic metadata are not distributed, the load balancing only considers *distributable* metadata (static and less dynamic metadata) on the gateways. We initially assign ranges to the gateways/nodes in the tree depending on the uniform distribution of the distributable metadata. But during the session run time, the distribution of the metadata can change and the load can be skewed. For example, some tree nodes may store large number of metadata while others have very few mapped in their ranges. Thus, load on tree nodes can be skewed. We use a *load balancer* similar to the Direct Neighbor Repeated [6] load-balancing approach. Each node periodically exchanges messages to its neighbor about its own load along with the heartbeat message and the load information is eventually propagated to the root. If the ratio of maximum load and minimum load in the system is greater than a threshold, the root initiates a load balancing algorithm where the highly loaded nodes transfer the load to its candidate range neighbors.

As range is a one-dimensional index, and nodes are assigned disjoint ranges, we can define node *adjacency* depending on their range values. One node is adjacent to other if

their ranges are consecutive regardless of their positions in the overlay tree. The node which has the highest range index is considered to be adjacent to the node with lowest range index. Hence, they form a continuous logical range ring as shown in Fig. 8. Node adjacency is further classified in terms of predecessor and successor. The predecessor and successor of a node along the ring are the nodes that have the subsequent ranges before and after the given node. If one node decides to shorten its range, the only candidate nodes that can take the remaining range are the nodes that are predecessor or successor of that node in the ring. Thus, the purpose of this ring is only to allow nodes to increase/decrease of the size of the range index assigned to them so that the number of metadata distributed to each node can be made balanced.

The load balancer works in several passes. Every node periodically reports the *minload* and *maxload* of its sub-tree to its parent via heartbeat messages. The root has the load information of the whole system. For an imbalance with $load > a \times minload$, the root initiates the load balancing algorithm. If any node in the path finds its $load > a \times minload$, it makes $load = \max(load/2, \alpha \times minload)$ and adjusts its range and data items accordingly. It adjusts its range and transfers the excessive data items to its successor or predecessor defined in the range ring. When the balancing reaches the *minload* node, it notifies the root and the current pass is done. After sometime, root starts another pass. We use $a=2$ so that no nodes are allowed to store data items more than double of other nodes. Figure 8 shows an example of load balancing operation.

6 Session adaptation

Supporting 3D teleimmersion is challenging due to the huge demand of networking and computing resources. The scalability of the systems is not in the number of environments, but in the streaming density scale among a small number of environments. This is a very different scalability model from traditional peer-to-peer media streaming applications. In 3D teleimmersive environments, a slight increase in the number of environments significantly increases the streaming density in the system, due to the large number of I/O devices present in each environment and the large amount of system/network resources each data stream requires. The challenge of scalability is partially addressed in the session initiation protocol (more details in Section 4) where the construction of an initial content dissemination overlay topology is considered.

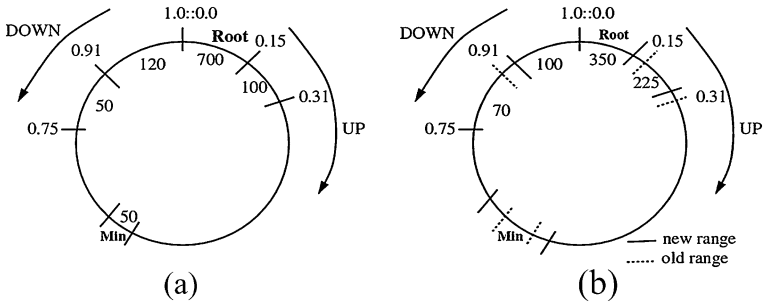

Fig. 8 Load balancing for $\alpha=2$: **a** initial load and, **b** after loads and ranges are adjusted (e.g., load 120 in **a** changes in **b** to new load $100 = \max(120/2, 2 \times 50)$, moving then 20 load units to the next interval $(50 + 20 = 70)$)

However, the session initiation protocol only assumes user view requests as static configuration and does not deal with the challenge of 3D interactive collaboration very well. During a teleimmersive session, it is quite often that users change their views which causes the stream membership change in multicast groups. Conceivably, over certain period of time the initial forest topology for 3D data dissemination will lose its efficiency and needs to be adjusted. Although it is possible to rerun the session initiation, the protocol executes at a global level and lacks resilience to tolerate frequent changes. Therefore, a runtime adaptation mechanism is needed to accommodate the dynamics of the system. The session adaptation will take the initial data dissemination topology as the starting point and gradually modify it as the system evolves. The goal includes: (a) to maintain the efficiency of content dissemination topology under the bandwidth and latency constraints, and (b) to alleviate the impact of the dynamic changes.

There are two types of session adaptation: (a) *application-driven* adaptation deals with configuration of resources for the optimal delivery of teleimmersive contents given the dynamic change of user interests; (b) *system-driven* adaptation on the other hand is triggered by the fluctuation of system resources such as the available bandwidth. In the current system, both types of adaptation are performed using view-based approach as instantiated by the ViewCast protocol [42]. The basic idea of ViewCast is to select data for transmission based on their semantic importance, and construct a dissemination topology adaptive to network and system dynamics (e.g., available bandwidth, end-to-end latency, etc.). More specifically, the semantic importance of the streams is dynamically determined with reference to the view selected by the user based on the view model. Only a subset of streams that is contributing to this view will be transmitted. A substrate component manages the stream selection, overlay construction, session management, and data adaptation among sites, making it transparent to the user. The substrate also keeps track of available bandwidth, point-to-point latency in the network, and resource usage of all service gateways. More importantly, it maintains an adaptive application-level overlay to deliver the semantically important streams to the users based on their selected views.

Figure 9 illustrates the general idea of how the ViewCast protocol works. Each node (i.e., v_1,v_2,v_3,v_4,v_5) represents a service gateway (SG). The service gateways form an overlay network and cooperate for content delivery. When a view change is detected, the new view request is sent to the local service gateway. Suppose user u_2 with its display and renderer

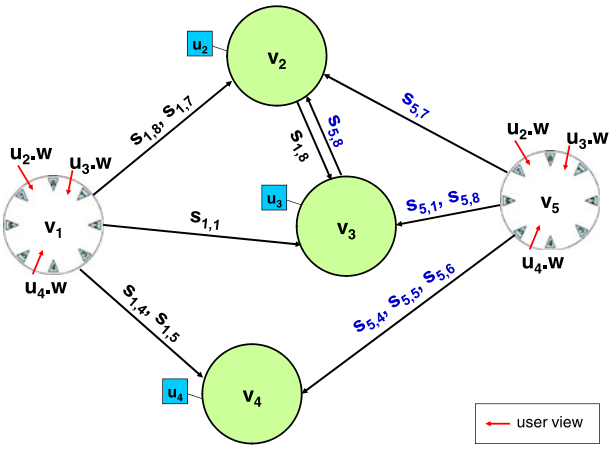

Fig. 9 ViewCast-based multi-streaming

registers with node v_2 and requests a view (denoted as $u_2.\bar{w}$) from node v_5. Through the stream differentiation, stream $s_{5,7}$ and $s_{5,8}$ are selected to serve the view. One of the streams ($s_{5,7}$) is transmitted directly from the source (i.e., v_5) while the other stream ($s_{5,8}$) is relayed from an intermediate node (i.e., v_3). As long as the quality and resource constraints are satisfied, the nodes other than the original source that have available streams can serve other nodes. Furthermore, a node can retrieve streams from multiple nodes in parallel. Note that, depending on the distribution of user views and available resources, similar user views may receive different set of streams.

ViewCast is an online protocol which dynamically adjusts the content dissemination topology. Depending on the application scenario, the protocol can either start with the initial content dissemination topology created in the session initiation (Section 4) or from scratch using a set of user requests as input. During a live 3DTI session, the user can switch his/her viewing position of the virtual space at the renderer. If the view change cannot be accommodated due to the lack of required streams, the renderer will forward the request to its local service gateway. The service gateway checks whether it has the streams available for serving the view change. If not, it sends a view request to the session controller to compute a new multicast topology for coordinating the multi-streaming. After a new dissemination topology is calculated, the session controller broadcasts it to all service gateways to complete the view request. More details about ViewCast message exchanging are provided in Section 6.3.

6.1 ViewCast goals

The solution of ViewCast has two major designing goals.

- *Minimum quality guarantee*. Each service gateway should receive a minimum set of streams to have some rendering quality guarantee of every other node inside the user view. For 3D teleimmersive interaction, it implies the consistent presence of all participants in the virtual space, which is critical for the collaborative work.
- *View change resilience*. When one user changes his/her view, the impact on other users should be minimized for the continuity of group interaction.

6.1.1 Minimum quality guarantee

The basic problem of creating a content delivery topology for satisfying minimum quality guarantee is NP-hard [41]. If we further consider the dynamics of the user view interests, the problem will become even more complicated. The solution of ViewCast relies on view-oriented heuristics. The user requests are first mapped into stream requests which are then selected and prioritized (Section 4.2.2). The streaming resources are then scheduled using the prioritization-based approach to make sure streams above certain minimum priority are delivered first. In addition, the resource used by lower priority stream can be *preempted* by higher priority stream under resource constraints.

The preemption mechanism is important in the resource-constrained 3D teleimmersive environments. Since bandwidth is a sparse resource, prioritization-based preemption is a key to guarantee efficient utilization of the resource in times of contention. Further, preemption is not intrusive (in the sense of visual quality disruption) due to the 3D data representation. All streams are aggregated and rendered into a single representation, which is very different from the conventional 2D video systems where losing a stream means losing a scene (e.g., the window showing that stream). Here when the user is observing the

scene from the front view, for example, it will not be as visually noticeable if some stream from the side view is preempted. The preempted resource can then be used to serve higher-priority streams which contribute more to the overall visual quality.

6.1.2 View change resilience

The possible negative impact of the view change is due to the fact that service gateways rely on each other for forwarding the content. As illustrated in Fig. 9, user u_3 and u_2 have similar views and u_3 is getting stream $s_{1,8}$ as relayed by service gateway v_2. Suppose u_2 changes its view, then gateway v_2 may stop receiving stream $s_{1,8}$ and forwarding it to u_3. In such case, u_3 has *victims* (i.e., broken streams). The impact will grow as the number of dependent nodes increases. The view change operation is a frequent phenomenon in 3D teleimmersive environments which needs to be well addressed.

We apply the strategy of *dependency balancing* to improve the view change resilience of the whole system. Three techniques are used including: (1) *source balancing*, (2) *priority balancing*, and (3) *forwarding load balancing*. Source balancing attempts to diversify the supplying nodes to lower the dependency on each individual node. Priority balancing attempts to provide a more even distribution of supplying nodes in terms of the priority of required streams. Forwarding load balancing attempts to balance the forwarding load among all the nodes.

6.2 ViewCast management

The main task of ViewCast management is to serve the view request, which is performed by the session controller and service gateways cooperatively. The main *ServeView* Algorithm 6.1 is sketched below.

Algorithm 6.1 *ServeView*(u, v_i)
  Input:
    u: a user who sends the view request;
    v_i: the service gateway which u belongs to;
begin
  $\bar{S} \leftarrow \emptyset$;
  for all environment S **do**
    $\bar{S} \leftarrow \bar{S} \cup get\_streams(S, u)$;
  end for
  for all stream s $\in \bar{S}$ in descending order of priority **do**
    $v \leftarrow find\_source(s, u)$;
    if (v found)
      $out \leftarrow allocate\_outbound\_bw(s, v, v_i)$;
      $in \leftarrow allocate\_inbound\_bw(s, v, v_i)$;
      if (*out* found **and** *in* found)
        $serve\_stream(s, v, v_i)$;
      end if
    end if
  end for
  $fix\_victim()$;
End

The *get_streams* routine calculates the differentiation function *df* with the given view. The selected streams are assigned priority and saved in *S*. The *find_source* routine searches for a supplying node, that can transmit stream s ($s \in s$) to the user, while achieving source balancing, priority balancing and forwarding load balancing. After a supplying node is located, the bandwidth is allocated for both outbound and inbound transmission. Once the allocation is successful, the supplying node can start streaming content to the requesting node. Note that, since stream preemption is applied, the aforementioned procedure may generate victims. The affected nodes are saved in the victim set and fixed by the *fix_victim* routine. More specifically, a new request is generated for each disrupted stream and served in a similar way as *ServeView*. To reduce the cost, this routine only fixes broken streams with high priority. For low priority streams, it simply ignores them and recursively processes the child nodes. The *fix_victim* routine will terminate either when the preempted stream is not important for all child nodes or some child nodes have found new sources. After the *ServeView* routine is completed, the session controller calculates the new content delivery topology and broadcasts to all nodes using the messaging protocol as discussed in the following section.

6.3 ViewCast messaging

Figure 10 illustrates the protocol starting from requesting a view to retrieving the contributing streams. At any given time, the streams transmitted always contribute the most to the selected view. Every view change made by the user may trigger a ViewCast request (VC_REQ in Fig. 10) from his/her renderer. Below we describe the basic steps in the protocol.

Step 1. Requesting Renderer→Local Gateway. On detecting a change of user view that cannot be accommodated with available streams, the renderer sends a VIEWCAST_REQUEST message to the local gateway. The message payload consists of a list of ViewCast request items, with each item representing a request for one 3D video stream. The renderer selects these streams by computing the semantic importance of each stream with reference to the user-selected rendering view. The message is filled up by the renderer and sent to the local gateway.

Step 2. Local Gateway→Session Controller. On receiving the VIEWCAST_REQUEST message from a requesting renderer, the local gateway fills in the first two entries in the message header: (a) gateway sequence number—a monotonically increased sequence number the gateway uses to keep track of its requests, and (b) gateway

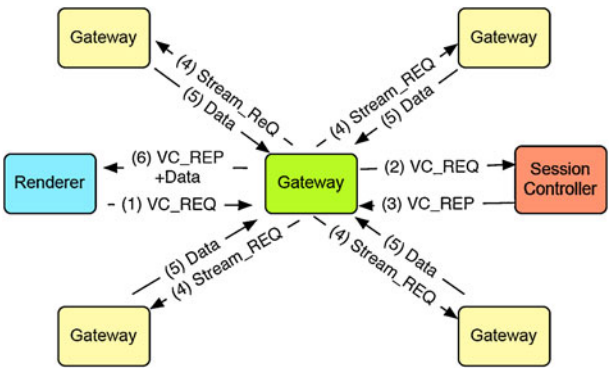

Fig. 10 ViewCast messaging

address—the IP address of itself. It then forwards this request message to the session controller. As mentioned earlier, the gateway only sends requests for foreign streams to the session controller.

Step 3. Session Controller→Local Gateway. On receiving the VIEWCAST_REQUEST message from the requesting gateway, the session controller sends a VIEWCAST_REPLY message back. The message payload consists of a number of ViewCast reply items, with each item representing a reply for one requested stream. The session controller generates these reply items by executing the *ServeView* routine (refer to Algorithm 6.2) which depends on the current topology and network/system dynamics to select appropriate parent nodes, if possible, to serve the requesting gateway. The message is filled up by the session controller and sent to the requesting gateway.

Step 4. Local Gateway→Remote Gateways. The requesting service gateway examines the VIEWCAST_REPLY message, and then sends a STREAM_REQUEST message to the parent gateway specified in each ViewCast reply item (refer to Fig. 10). The STREAM_REQUEST message contains the stream name and the IP address of its requesting gateway.

Step 5. Remote Gateway→Local Gateway. The remote service gateway resolves the STREAM_REQUEST upon its reception by sending the requested stream to the requesting service gateway.

Step 6. Local Gateway→Requesting Renderer. Receiving the stream from the parent node, the local service gateway distributes this stream to the requesting renderer, together with a VIEWCAST_REPLY message. To this point, the ViewCast request is resolved.

6.4 Transmission overlay management by session controller

The session controller maintains three types of information: (a) I/O devices and gateways membership, (b) network and system dynamics, and (c) 3D data transmission overlay topology. For membership, it acquires a list of gateway nodes, and a list of participating cameras and displays via the session initialization protocol. It also keeps track of the network and system dynamics such as point-to-point latency and end-to-end available bandwidth between pairs of gateway nodes. Such dynamic information is needed for the 3D data transmission overlay construction. Most importantly, the session controller maintains the 3D data transmission overlay topology, i.e., how the gateway are connected on the application-level overlay for stream dissemination.

7 Session termination

The session termination protocol deals with gracefully tearing down various components of the teleimmersive session including cameras, displays and service gateways. A 3D camera unit leaves the system by sending the message of CAMERA_LEAVE to its gateway. Due to the synchronization mechanism, once a camera starts to leave, the gateway needs to turn off all the rest cameras. After that, the gateway forwards a message of GATEWAY_UPDATE to the session controller. The session controller then propagates the message to other gateways to release the resources related with streaming 3D content originated from that gateway. At this moment, the gateway does not generate any content of its own, but can still remain in the system to serve its displays and help other gateways.

A display leave is handled in a similar way by sending the message of DISPLAY_LEAVE to the gateway. In case there is no other display attached to it, the gateway forwards a message of GATEWAY_UPDATE to the session controller. The current content delivery topology needs to be adjusted since the gateway no longer needs to receive content for its own usage. However, there could be other gateways that are currently relying on it so the session controller will continue to use the gateway as content relaying node to avoid significant topology adjustment. Once such dependency is released, the session controller will no longer consider it as a potential helping node to reduce the level of problem complexity. The gateway, on the other hand, can still stream its own content to other gateways if it has cameras attached to it.

The gateway leaves the system by sending the message of GATEWAY_LEAVE to the session controller. Under such event, the gateway no longer outputs its content to the session, or receives and relays contents related to other environments. The gateway can still have its own cameras and displays running, but its environment is separated from the rest of the session. Such leave could be temporary (e.g., for certain privacy concern) and the gateway can rejoin the session later. Currently, more advanced protocols are being developed that deal with the *multi-session* concept where several environments can form a session and jointly separate from or merge with other on-going sessions involving other environments.

Recall that, the information kept in the 3D teleimmersive session remains soft-state for reliability concerns. Therefore, the messaging related to session termination provides only an explicit way to tear-down the session. There are detection mechanisms to deal with the case when certain components leave abruptly (refer to Section 5.5). Once detected, those events will be converted into appropriate messaging for proper handling. For example, a camera freeze can be detected with the session monitoring protocol. The gateway will then turn off rest cameras and send a message of GATEWAY_UPDATE to the session controller.

8 Related work

We will review the related work in several aspects. We will examine session related real-time transport protocols, tele-conferencing systems on the current market, and the teleimmersive systems under development.

8.1 Session and real-time transport protocols

Session Initiation Protocol (SIP) is an application layer signaling protocol defined in [22] and is mainly used for session management in IP Telephony and VoIP services. The end users use SIP to initiate a session, invite other users, add/drop participants to the session, and finally terminate a session. SIP supports on-the-fly modification of the sessions and supports both unicast and multicast sessions between the users. SIP is only a signaling protocol and hence, does not provide mechanism to transfer the actual data or the session information. For this, protocols like RTP (Real-time Transport Protocol) are used for transferring the data and SDP (Session Description Protocol) for transferring the session management information. SIP provides much functionality for session management of IP Telephony type of services however; it does not support more complex features for advanced multimedia systems like teleimmersion dealing with heterogeneous streams (local streams vs. remote streams), requiring monitoring of resource allocation, and building not just unicast/multicast but more complex topologies between the users. SIP has no notion of

describing correlation between the streams which is an important feature of correlated 3D streams in 3D teleimmersion systems.

RTSP, the Real Time Streaming Protocol, is a client–server multimedia presentation protocol to enable controlled delivery of streamed multimedia data over IP network [21]. It provides "VCR-style" remote control functionality for audio and video streams, like pause, fast forward, reverse, and absolute positioning. RTSP can be used for transferring the control signals however; it relies on RTP and underlying transport layer protocols to stream the actual data. Some applications using RTSP are QuickTime Streaming Server, VideoLAN, FFmpeg, Windows Media Services, MPlayer, RealPlayer, Skype, Winamp, etc. RTSP has been successful in controlling videos streamed on web however; it lacks support for more complex control like users changing 3D views in 3D teleimmersive systems. To support these requirements of the emerging multimedia systems, there is an urgent need of more advanced session initiation and session control protocols.

H.323 is another session initiation and management protocol which is an entire suite of protocols providing functionalities ranging from call control, conferencing, and codecs to many other features [12]. The advantage of using H.323 is that it provides large amount of functionality in itself and hence, decreases the dependence on protocols like RTP, SDP, and RTSP. This in fact increases the overall performance of the protocol. However, in the world of ever changing IP telephony requirements, H.323 becomes lesser flexible and difficult to modify for emerging multi-stream multi-modal systems.

As it has been discussed so far, most of the applications and protocols depend on Real-time Transport Protocol (RTP), a well-known application layer protocol to transfer individual real-time media streams [23]. RTP provides time stamping for synchronization, loss detection, security, content and source identification. A separate RTP session is initiated for each multimedia stream and uses *Application Level Framing* and *Profiles and Payload Formats* to define the application headers for adding new multimedia formats, and specify codecs, and the corresponding multimedia encoded data respectively. RTP is used in conjunction with Real Time Control Protocol (RTCP) which is discussed in further detail in the next paragraph. RTP can use any underlying transport layer protocol however, majority of RTP implementations use UDP to avoid delays. Since RTP leaves a lot of design hooks open for the specific needs of different applications, an enhanced version of it can be designed to work with teleimmersive streams.

The Real Time Control Protocol (RTCP) standardized in [23] is an application layer protocol which works together with RTP. RTCP provides an out-of-band control channel for transmitting control information about lost packets, jitter, and delay. Based on the monitoring statistics, the participants can adaptively change the coding technique, drop lower priority packets and detect faults. Each end host is given a unique source name called canonical name to allow adaptations in the middle of the session. The traffic introduced by the RTCP control signals is kept minimal so as not to inhibit transfer of actual data. Although RTCP provides functionalities for monitoring several metrics however, there is a need for a different monitoring plane which can handle multi-attribute range metrics like for a given stream rate find total CPU, bandwidth, and memory overhead at each node and then optimize the frame rate based on the results.

8.2 Tele-conferencing systems

There is a number of video and audio conferencing systems on the current market. We classify these systems in Fig. 11.

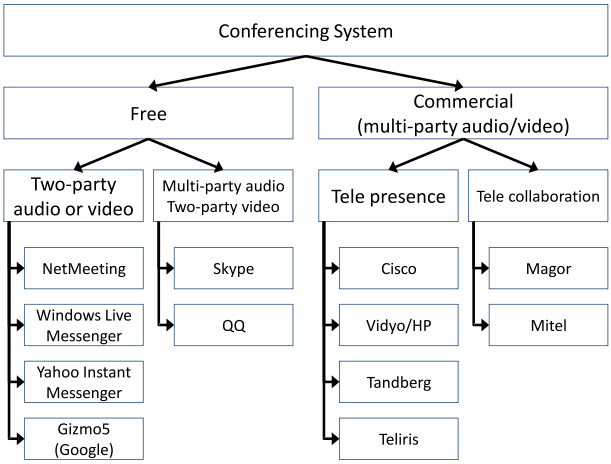

Fig. 11 Classification of multi-media conferencing systems

All free conferencing systems can support two-party audio or video conference. Microsoft NetMeeting [17] used H.323 protocol and it also used a proprietary session protocol known as Microsoft Audio Call Control Protocol [18]. However, it has embedded SIP into its.Net framework which Windows Live Messenger [34] is built upon. Both Yahoo messenger [43] and Gizmo5 (acquired by Google) [9] uses SIP for their voice connection. Skype [27] and Asian-based QQ [19] also support SIP, and can provide multi-party audio service (but video conference is limited to two-party).

Skype (v3.5) [24] uses peer-to-peer connections for two-party audio and video conference, and a centralized scheme for multi-party audio. At the stage of session initialization, Skype decides the appropriate transport protocol. It prefers UDP for audio transmission and TCP as a backup when UDP packets cannot get through due to a possible Firewall issue. When both protocols fail, Skype establishes connections by relaying packets through a Skype supernode. In the multi-party audio, Skype selects the person who starts the multi-party conference as the central node. The central node mixes and forwards the audio signals to other parties.

The commercial systems on the market usually support multi-party videos. These systems can be broadly divided into two categories: telepresence [5, 29, 30, 32] and telecollaboration [15, 16]. H.323 is widely supported in both categories. Depending on the specific systems, SIP may or may not be supported. Conventional telepresence systems use the multi-point control unit (MCU) solution which may not function well under Internet, screen size and processing power heterogeneity. In addition, adding additional participants to MCU can be difficult. Vidyo [32] however claims its patent-protected telepresence solution (which still requires a central server) eliminates MCU and hence the aggregate latency and its H.264/Scalable Video Coding solution can better adapt to Internet bandwidth variation. Teliris [30] also proposes its rate adaptation technology in its 6 G telepresence platform.

Contrary to telepresence, telecollaboration manufacturers like Magor [15] and Mitel [16] adopt a peer-to-peer communication architecture which allows the end users to decide what they want to see. Adding and removing participants in telecollaboration can become much easier compared to telepresence centralized scheme. The end-to-end latency between two peers can be reduced due to the removal of MCU or central server in telepresence. The video quality now depends on the bandwidth availability between every two peers in the system.

8.3 Teleimmersive systems

In the teleimmersion system developed at UNC Chapel Hill, a coordination protocol is proposed to monitor multi-stream traffic between two ends [20]. It is a transport layer protocol which specifies how a set of correlated streams registers with the last-hop router, how the router collects bandwidth usage information from them, and how the applications query the router. The protocol provides a profiling channel but the actual task of traffic control is left to the teleimmersive applications. Thus, it only serves as a component of lower layer protocol for session management, not to mention that it is restricted to two-party communication.

The work of Coliseum illustrates a teleimmersive system on desktop with each end having five cameras mounted in a semi-circle around the above-shoulder area [4]. The 3D view of a user is captured simultaneously by those cameras, extracted from background, reconstructed and embedded into the virtual environment. One significant difference between Coliseum and other teleimmersive systems is that instead of transmitting multiple 3D streams each end creates a single 2D stream by rendering the 3D scene locally and then transmits it to the remote end. From that sense, Coliseum resembles more a traditional single-stream multi-party tele-conferencing system. Discussion on session management is very limited.

In contrast, the blue-c project represents high-end immersive systems with advanced projection technology for large spatial displays [11]. The video system installs 16 cameras covering 360° of a CAVE-like environment. During runtime, 3 cameras are selected for the texture and 5 cameras for 3D reconstruction according to the user view. However, the work is more focused on video and rendering portion for optimal visual quality. The issues of QoS adaptation according to the user requirement and available bandwidth, related spatial and temporal quality trade-off, and the coordination of multiple parties are not addressed.

The TEEVE system at UIUC supports multi-camera/multi-stream 3D video capturing and transmitting, and multi-party immersive communication [38]. By far, the work provides the most comprehensive study on the session management for teleimmersive systems with various aspects including multi-party membership management, cross-stream coordination, content overlay construction, dynamic QoS quality adaptation, and performance monitoring [2, 36, 40–42].

9 Conclusion

In this paper, we highlight several important aspects of session management for teleimmersive systems as observed in several years of development. Currently, systems equipped with high-quality camera and display devices are moving towards supporting real large-scale multi-party teleimmersive applications exemplified by collaborative dancing [25], rehabilitation training and study [3], and Tai-chi learning [14].

As teleimmersive systems start to exert social and scientific impacts, those applications bring up challenging demands such as quick set up, easy operation by non-technical personnel, reliable connectivity and automatic quality and user experience adaptation. To address those challenges, we should set the goal of making tomorrow's teleimmersive systems as simple to use as today's telephone system. However, it is also a trend that teleimmersive systems are becoming more complicated at a rapid pace with the advancement of media devices (e.g., audio, visual, and haptics, etc.). Now is a critical point that it becomes really difficult to maintain teleimmersive sessions with ad-hoc or technician-intensive solutions. The needs for the next generation session management and its new session initiation, monitoring and adaptation protocols are very clear if we want to extend the scope of teleimmersive applications. However, in contrast to the substantial progress made in the capturing and

rendering techniques, the research of session management is lagging behind as reflected in the literature and available systems. Today, most systems still rely on single-stream end-to-end transport-layer protocols. It is hard to adapt those protocols to take into account application-layer semantics and perform cross-stream multi-party transmission control. As presented, those desirable features of session management are urgently needed by teleimmersive systems but are still under-investigated. Very soon, they will pose big barriers for the sustainable development of these environments. We thus strongly believe that new research efforts in the study of designing appropriate session protocols and their standardization for 3D teleimmersive interactive systems are necessary and our presented design is the first step in this direction.

References

1. Agrawal M, Davis L (2002) Trinocular stereo using shortest path and the ordering constraint. Int J Comput Vision 47:43–50
2. Arefin A, Sarwar Y, Gupta I, Nahrstedt K (2009) "Q-Tree: a multi-attribute based range query solution for teleimmersive framework," IEEE International Conference on Distributed Computing Systems
3. Bajcsy P, McHenry K, Na H-J, Malik R, Spencer A, Lee S-K, Kooper R, Frogley M (2009) "Immersive environments for rehabilitation activities", In the Proceedings of ACM International Conference on Multimedia, Beijing, China, October 19–24
4. Baker H, Bhatti N, Tanguay D, Sobel I, Gelb D, Goss M, Culbertson W, Malzbender T (2005) "Understanding performance in coliseum, an immersive videoconferencing system," ACM Transactions on Multimedia Computing, Communications, and Applications
5. Cisco Telepresence, http://www.cisco.com/telepresence
6. Corradi A, Leonardi L, Zambonelli F (1999) "Diffusive load-balancing policies for dynamic applications," In the Proceedings of IEEE Concurrency, vol. 7, no. 1
7. Dabek F, Cox R, Kaashoek F, Morris R (2004) "Vivaldi: a decentralized network coordinate system", In the Proceedings of the ACM SIGCOMM '04 Conference
8. Daniilidis F, Mulligan J, Mckendall R, Majumder A, Kamberova G, Schid D, Bajcsy R, Fuchs H (1999) "Towards the Holodeck: an initial testbed for real-time 3D teleimmersion", ACM SIGGRAPH
9. Gizmo5, http://www.gizmo5.com
10. Goemans M (2003) "Minimum bounded degree spanning trees" In the Proceedings of IEEE Symposium on Foundations of Computer Science, pp. 273–282
11. Gross M, Würmlin S, Naef M, Lamboray E, Spagno C, Kunz A, Koller-Meier E, Svoboda T, Gool LV, Lang S, Strehlke K, Moere AV, Staadt O (2003)"Blue-c: a spatially immersive display and 3d video portal for telepresence," ACM Trans Graph
12. Illinois Teleimmersion, http://cairo.cs.uiuc.edu/teleimmersion
13. International Telecommunication Union (2003) "Packet based multimedia communication systems," Recommendation H.323
14. Jain M, Dovrolis C (2002) "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput," In the Proceedings of ACM SIGCOMM, pages 295–308
15. Kurillo G, Vasudevan R, Lobaton E, Bajcsy R (2008) "A framework for collaborative real-time 3D teleimmersion in a geographically distributed environment," In the Proceedings of the 10th IEEE International Symposium on Multimedia
16. Magor Telecollaboration, http://www.magorcorp.com
17. Microsoft NetMeeting Protocol Specification, Microsoft Corp., June 2010
18. Microsoft NetMeeting, http://www.microsoft.com
19. Mitel Telecollaboration, http://www.mitel.com
20. Ott DE, Mayer-Patel K (2004) "Coordinated multi-streaming for 3d teleimmersion", In the Proceedings of the 12th annual ACM International Conference on Multimedia
21. QQ Messenger. http://www.qq.com
22. Rosenberg J, Schulzrinne H, Camarillo G, Johnston A, Peterson J, Sparks R, Handley M Schooler E (2002) "SIP: Session Initiation Protocol (RFC 3261)"
23. Sat B, Huang Z, Wah BW (2007) "The design of a multi-party VoIP conferencing system over the internet," In the Proceedings of IEEE International Symposium on Multimedia

24. Schulzrinne H, Casner S, Frederick R, Jacobson V (2003) "RTP: A transport protocol for real-time applications (RFC 3550)"
25. Schulzrinne H, Rao A, Lanphier R (1998) "Real time streaming protocol (RFC 2326)"
26. Sheppard R, Kamali M, Rivas R, Tamai M, Yang Z, Wu W, Nahrstedt K (2008) "Distributed virtual collaboration through Teleimmersive Dance (TED): A symbiotic creativity and design environment for art and computer science," In the Proceedings of ACM International Conference on Multimedia, Vancouver, BC, Canada
27. Shi S, Nahrstedt K, Campbell RH (2008) "View-dependent real-time 3D video compression for mobile devices" In the Proceedings of ACM International Conference on Multimedia
28. Skype, http://www.skype.com
29. Svoboda T, Martinec D, Pajdla T (2005) "A convenient multi-camera self-calibration for virtual environments," In PRESENCE: Teleoperators and Virtual Environments 14:407–422
30. Tandberg Telepresence, http://www.tandberg.com
31. Teliris Telepresence, http://www.teliris.com
32. Vidyo Telepresence, http://www.vidyo.com
33. Wang Z, Crowcroft J (1996) Qos routing for supporting resource reservation. IEEE J Sel Areas Commun 14:1228–1234
34. Windows Live Messenger, http://messenger.live.com.
35. Wu W, Yang Z, Gupta I, Nahrstedt K (2008) "Towards multi-site collaboration in 3D teleimmersive environments" IEEE International Conference on Distributed Computing Systems
36. Wu W, Yang Z, Nahrstedt K (2008) "Implementing a distributed 3D Teleimmersive System," In the Proceedings of the 10th IEEE International Symposium on Multimedia, Berkeley, CA, USA, December 15–17
37. Wu W, Yang Z, Nahrstedt K (2008) "A study of visual context representation and control for remote sport learning tasks", AACE World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA), Vienna, Austria
38. Yahoo Instant Messenger, http://messenger.yahoo.com
39. Yang Z, Cui Y, Yu B, Liang J, Nahrstedt K, Jung S-H, Bajcsy R (2005) "TEEVE: the next generation architecture for teleimmersive environments," In the Proceedings of the 7th IEEE International Symposium on Multimedia, Irvine, CA, USA
40. Yang Z, Wu W, Nahrstedt K, Kurillo G, Bajcsy R (2007) "Viewcast: view dissemination and management for multi-party 3d teleimmersive environments," In the Proceedings of the 15th ACM International Conference on Multimedia
41. Yang Z, Wu W, Nahrstedt K, Kurillo G, Bajcsy R (2010) "Enabling multi-party 3D teleimmersive environments with ViewCast," ACM Transactions on Multimedia Computing, Communications, and Applications, vol. 6, no. 2
42. Z. Yang, B. Yu, K. Nahrstedt and R. Bajcsy, "A multi-stream adaptation framework for bandwidth management in 3D teleimmersion," In the Proceedings of the 16th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), 2006
43. Yang Z, Yu B, Wu W, Danikov R, Nahrstedt K, Bajcsy R (2006) "Study of collaborative dancing in tele-immersive environment", IEEE International Symposium on Multimedia (ISM) 2006, San Diego, USA

Klara Nahrstedt is the Ralph M. and Catherine V. Fisher Professor at the Department of Computer Science, University of Illinois at Urbana-Champaign. Her research interests are directed towards multimedia systems, networks and applications including quality of service (QoS) support, real-time protocols, network management, guaranteed services, QoS in wireless networks, QoS-aware resource management, soft real-

time scheduling, QoS languages and translations, QoS-aware middleware, multimedia distributed systems and applications, 3D Tele-immersion, tools for creative choreography, multimedia and Internet security, quality of protection, wireless security in first responder systems, and security in power grid SCADA networks. She is the coauthor of the widely used multimedia book "Multimedia: Computing, Communications and Applications" published by Prentice Hall, the recipient of the Early NSF Career Award, the Junior Xerox Award, the IEEE Communication Society Leonard Abraham Award for Research Achievements, and the IEEE fellow.

Klara Nahrstedt received her B.A. in mathematics from Humboldt University, Berlin, in 1984, and M.Sc. degree in numerical analysis from the same university in 1985. She was a research scientist in the Institute for Informatik in Berlin until 1990. In 1995 she received her Ph.D. from the University of Pennsylvania in the Department of Computer and Information Science.

Zhenyu Yang joined the multimedia operating systems and networking group in the Department of Computer Science, University of Illinois at Urbana-Champaign under the supervision of Professor Klara Nahrstedt in Spring 2004, and received Ph.D. (thesis focus: 3D tele-immersive environments) in October 2007. After graduation, he continued working with Professor Klara Nahrstedt as a postdoctoral fellow until August 2008. Zhenyu Yang received his M.S. (thesis focus: fault tolerance and reliability) from the Department of Computer Science, University of Illinois at Urbana-Champaign, December 2002 and B.E. from the Department of Computer Science and Engineering, Shanghai Jiao Tong University (CHINA), July 1994. His research interests include networking and distributed systems with a focus on communication and multimedia systems involving quality-of-service management, overlay and peer-to-peer networks, content delivery network, wireless networks, 3D collaborative environments, ubiquitous computing, human-computer interface and interaction, and the development of multimedia applications particularly to promote collaborative and interdisciplinary research.

Wanmin Wu is a Ph.D. student in computer science at University of Illinois at Urbana-Champaign. She received her B.S. degree in computer science from Zhejiang University, China. Her work focuses on human-centric designs and the way they drive 3D tele-immersive systems. Her research has been recognized by

multiple awards including Second Place in ACM Student Research Competition 2007, Yahoo! Key Technical Challenges Grant 2008, IBM Watson Emerging Leaders in Multimedia 2008, and Yee Memorial Fellowship 2010.

Ahsan Arefin is a Ph.D. student in Computer Science at the University of Illinois at Urbana Champaign. His research interests include Management and Analysis of Large-Scale Distributed Systems focusing Distributed Multimedia Systems, Quality of Service Management, Large-scale Monitoring and Data Mining. He is currently a member of SIGMM educational committee. Ahsan Arefin received his B.Sc. in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET), Bangladesh in 2006. He received various scholarships including merit list from Government of Bangladesh and awards for programming and design competitions during his graduate and undergraduate studies.

Raoul Rivas is a Ph.D student in Computer Science at the University of Illinois at Urbana-Champaign. His research interests are in the area of Operating Systems and Multimedia Systems including Virtualization Techniques, Quality of Service, Soft-Real-time Scheduling, Power Management and Heterogeneous Architectures.

He has received various scholarships including a merit based scholarship from Secretaría de Educación (government board of education) in Mexico and the Intel Undergraduate Research Scholarship in 2006.

Raoul Rivas received a B.S. in Computer Science from the University of Illinois at Urbana-Champaign in 2007, and a B.S in Systems Engineering from Universidad Panamericana in Mexico City, Mexico in 2005.