

# **NETWORKING FOR IMMERSIVE TELEPRESENCE: ARCHITECTURES AND PROTOCOLS - A CASE STUDY**

*Sriram Sethuraman*

Interactive Media Group, Vision Technologies,  
Sarnoff Corporation, CN5300, Princeton, NJ 08543-5300  
Email: ssethuraman@sarnoff.com

## **ABSTRACT**

Immersive telepresence allows an observer to view a remote scene from any viewpoint of choice and thus give a sense of presence, as opposed to conventional video viewing where the user can only view content from the viewpoint of a single camera. We specifically consider a depth-based rendering approach for enabling telepresence. In this paper, we explore various networking architectures in which telepresence can be enabled over the Internet using this approach. From these architectures, we derive a set of requirements for describing and conducting a telepresence session. Based on these requirements, we present a session protocol that draws upon the features of RTSP and SDP and extends them. Various media streams from a single camera viewpoint are aggregated as a view group and the concept of this view group is used to support the various architectures. Both unicast and multicast configurations, with central or distributed servers, are supported by this protocol. Finally, we present the overall end-to-end architecture used in our current implementation.

## **1. INTRODUCTION**

Immersive telepresence requires a remote scene to be rendered from the viewpoint of an observer. Unlike traditional rendering methods where the geometry and the texture of the scenes are modeled, image-based rendering (IBR) methods [6, 7, 8] synthesize arbitrary views of a scene from a collection of images observed from known viewpoints (poses). When extended to dynamic scenes, the main advantage of the IBR approach is that it can render complicated scenes that are otherwise difficult to model geometrically in real-time. In this paper, we consider the depth-based rendering (DBR) approach [8, 9], where a 3-D dynamic scene is represented by:

- (a) Video streams from distributed cameras covering a portion of the 3-D space,
- (b) Static/dynamic estimated pose information for each camera, and
- (c) Dense, dynamic depth maps from the viewpoint of each camera at each time instant.

We refer to the system that generates the above representation for a given camera as a 3D-camera unit (3DCU). Hence, immersive telepresence over the Internet requires the transmission of the 3D representation from one or more 3DCUs. The representations from multiple viewpoints are combined at the client-side using the pose information to render the scene as observed from a desired viewpoint. In this paper, we consider the networking aspects of realizing such an application.

IETF's Real-Time Streaming Protocol (RTSP) [1] in conjunction with the Session Description Protocol (SDP) [2] enables complex session setups and client/server interactivity using an out-of-band control channel. The different media streams that are needed for a presentation can be located on multiple IP nodes. A control channel can be individually assigned to each media stream. RTSP also allows unique states for each session for each ongoing session to be maintained through *session\_id*. These features make these protocols a good choice for designing an immersive telepresence application. The specific requirements for the application in its potential modalities, however, require a few extensions to these protocols.

In Section 2, we present the various potential modalities in which immersive telepresence over the Internet is possible along with the merits and demerits of each modality. Specific requirements are derived from these modalities in Section 3. The concept of a group of media streams associated with each 3D-camera unit (called the view group) is introduced. The proposed protocol that draws upon the features of RTSP and SDP while meeting the application requirements is presented in Section 4. The overall end-to-end architecture and implementation specific details are provided in Section 5.

## 2. POTENTIAL NETWORKING MODALITIES

The requirements for the network architecture and protocol can be derived once the potential networking modalities for the chosen application and their ramifications are understood. It also provides valuable hints to deployment solutions. The primarily useful modalities for the DBR based immersive telepresence approach can be summarized as the following categories.

1. Streaming a pre-captured session in *unicast* according to the observer's dynamic viewpoint, where
  - a) the session streams are all available on one server
  - b) the session streams are distributed on multiple servers (say, due to storage requirements)
2. Streaming a pre-captured or a live *multicast* session with one or more network nodes (e.g. one node per 3DCU).

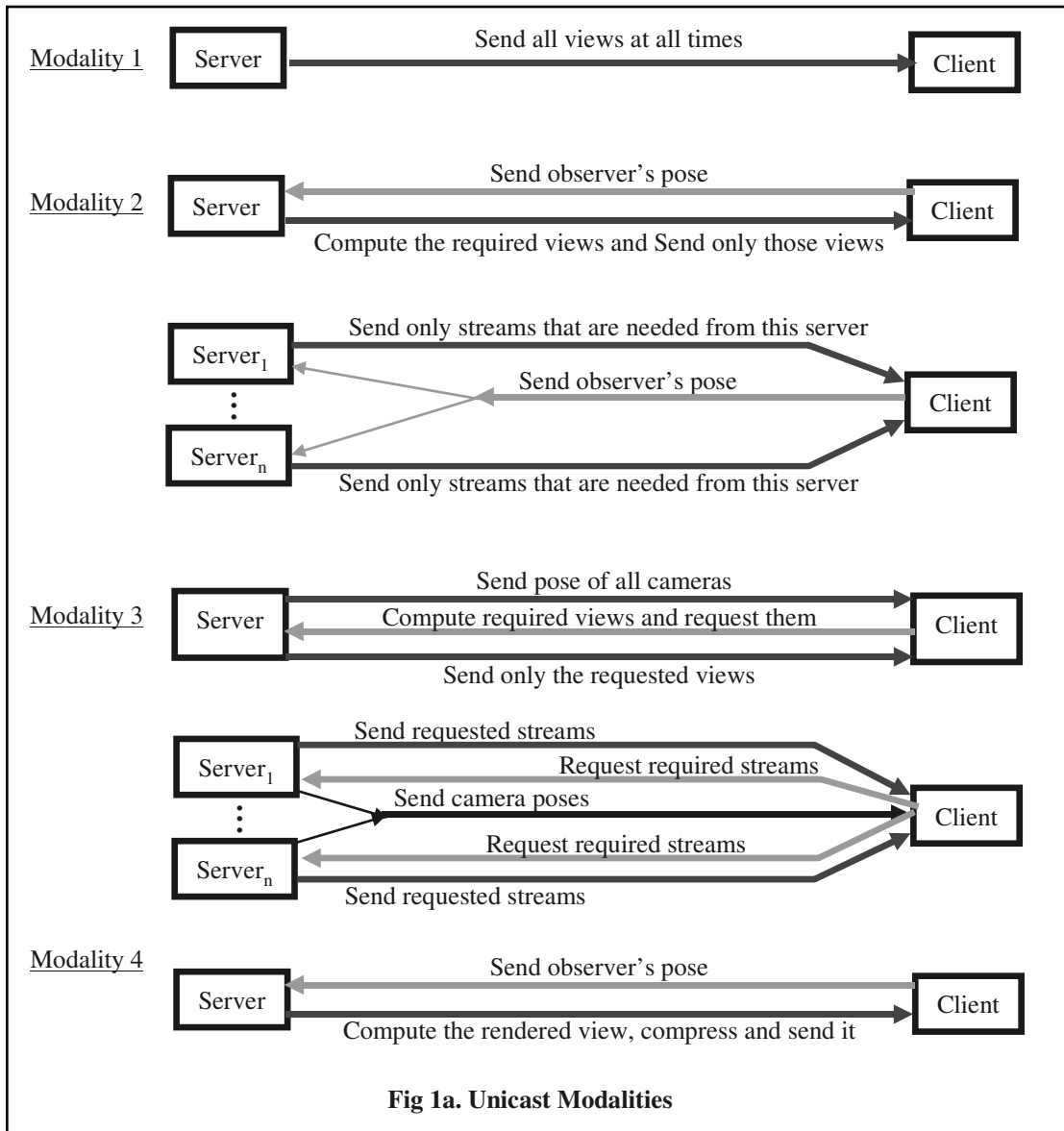
Unicasting enables on-demand applications where different users could individually interact with the server. Streaming a pre-captured session in unicast mode enables navigation in both space and time, while multicast mode can only allow navigation in space (or, at best, time navigation is limited to small durations due to buffering requirements at the client). The navigation in space is achieved through the 3D representation acquired from multiple 3D-camera units as explained in Section 1.

### Unicast mode

In the unicast mode, there are four possible modalities, namely,

1. Server sends all the camera views at all times
2. Server streams only the required camera views
3. Client requests the required camera views
4. Server composes the rendered view

These modalities are illustrated in Figure 1a.



The first modality is not very useful when the number of camera views available is high, as the aggregate bandwidth needed to send all the views would swamp the client application. It is for this reason that live unicast, which is also a possible modality, is not considered as a serious modality.

The second and third modalities are similar in the sense that only the required camera views for the current viewpoint of the observer needs to be sent from the server. In most typical scenarios with proper camera coverage, two camera views are sufficient to render a high quality virtual view. Hence, the bandwidth requirements are very modest when compared to the first modality. But these two modalities differ in whether the server or the client does the determination as to which views are required. As each 3D-camera unit itself could be moving, the pose information can change dynamically. If the server does the determination, then the observer's pose information needs to be sent to the server. In this case, the pose information for only the selected views needs to be sent to the

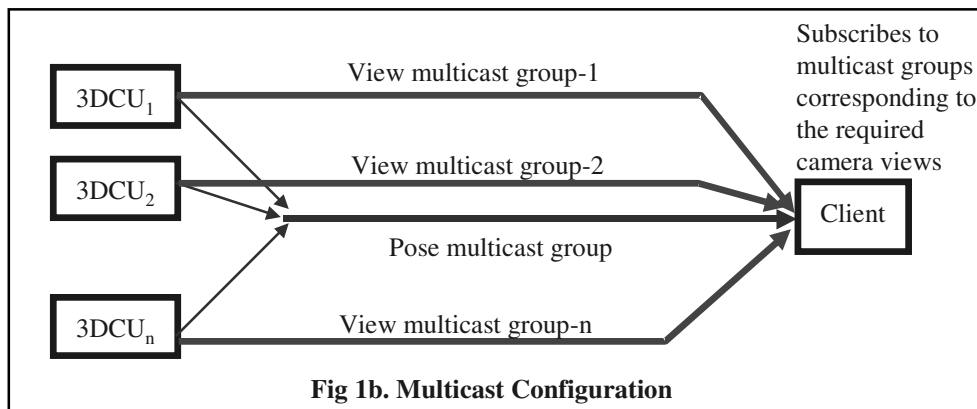
client to facilitate DBR. The drawback is that the server needs to spend cycles computing the required views. But, the advantage is that the server has a snapshot of all the poses at a given time. If the client does the determination, then it needs the pose information for all the 3D camera units covering the scene. In this case, due to packet loss (if pose is sent over UDP) or jitter, the client may not have the correct camera poses for a given time and may hence choose a wrong coverage. The rule we would use is that for simple camera modalities (e.g. fixed 3DCUs) or with limited sessions from a server, the server-side determination would be used. For complex modalities or with high number of sessions from server, client-side determination would be used.

In modality 2, when the session streams are distributed, the viewer’s pose needs to be sent to each server and each server should in turn serve only the required streams. For modality 3, the responsibility falls on the client to request streams from a subset of the servers at any given time and to inform other servers to cease transmission of streams that are no longer needed.

The fourth modality can be useful in cases where there are a small number of distributed browsers with very limited compute power or bandwidth. In this case, the observer’s viewpoint is sent to the server and the server performs DBR. This rendered view is then compressed and sent to that observer. Since only the final rendered view is sent, the bandwidth needed is also lower when compared to even sending compressed representations from two 3DCUs. Since the server needs to render the views for each client, this solution is not a scalable one. However, it may be ideal for an edge server where the last mile bandwidth is limited.

Multicast mode

In the multicast scenario, whether streaming live or from pre-captured sessions, the server does not respond to specific requests from each client. Though it is possible to collect receiver reports to turn off streams from 3DCUs that are not needed by any client at a given time, we do not consider this case because it would then not allow seamless navigation in 3D space. Given the bandwidth considerations from the client side, a good design in this case would be to assign a distinct multicast group for each 3DCU. Thus all the streams from a 3DCU (video, depth maps, and optionally audio) would be streamed on a single multicast group that we call as the view multicast group. This would allow client-side adaptation to subscribe to only the multicast groups that carry the views needed for the current viewpoint of the observer. Such a modality also allows distributed 3DCUs to be on different IP nodes. However, as mentioned before, for the client to make this determination, the pose information from all 3DCUs needs to be available at the client. Hence for dynamic 3DCUs, pose information from all 3DCUs can be sent together in their own multicast group. Figure 1b illustrates this configuration.



### 3. REQUIREMENTS FOR A SESSION

Given the various modalities for a typical DBR-telepresence session, we consider a session protocol for the unicast modalities 2 and 3, and the multicast modality. Unicast modality 4 can be easily achieved with any existing protocol such as RTSP as it involves streaming only one video stream. The session requirements that can be abstracted from the discussion in section 2 are:

1. The streams that are part of a single telepresence session can be on one or more IP nodes. Before the session could begin, the location of these streams has to be conveyed to the client (this functionality is already provided by SDP).
2. The concept of a *view group* is fundamental to associate the video, (audio), pose, and depth maps from a given 3DCU. This concept can be used either to request a specific set of view groups or to select the granularity at which to create a multicast group. Though RTSP can provide aggregated control channels, it does not provide a subset of media streams to be controlled by a single control channel.
3. In the unicast case, only a very small subset of actual number of view groups is sent from the server at any given time. To avoid having to connect and disconnect from specific RTP ports assigned to a stream, it might be better to have session-specific ports that are re-used by the active streams. The maximum number of simultaneous view groups needed for rendering a virtual view needs to be communicated to the server at the beginning of a session. There is no current provision in RTSP/SDP to specify re-use of port numbers.
4. The knowledge of the observer's pose at the server (when the server makes the determination about the set of views) or the dynamic poses of all the 3DCUs at the client (when the client makes the determination as to which views to request or subscribe) is critical. Since pose is also very compact (a 3x3 rotation matrix and a 3x1 translation vector), it may be worthwhile to send pose over TCP.
5. For seamless navigation in space and time, the file formats for video, audio, and depth should support random access. This would allow a new view that is requested to start very close to the current viewing time. It would also support seeking in time (fast-forward and fast-reverse).

### 4. SESSION PROTOCOL FOR DBR TELEPRESENCE

Based on the requirements in Section 3, we propose a new session protocol that is largely based on RTSP and SDP. We will use *trtsp* and *tsdp* to denote the telepresence extension to these protocols. The main modification is the fact that a view group serves as an aggregator for all media streams from one 3DCU. Hence it can be thought that this new protocol is like a 2-layer hierarchical version of RTSP and SDP. Though each media stream within a view group itself could be located on distributed servers and have independent controls, given the nature of the DBR application, we require all media streams from a 3DCU to be on a single server and to have a unified control.

The elements of this new protocol tailored for DBR telepresence are presented below:

1. Session request: The client starts with a TRTSP URL and requests a session description using the DESCRIBE command. Each session has a unique session name and each active session from a server will have a unique session ID.
2. Session Description: The server will respond by sending a container file that describes the session as follows:

```

s=<session name>
o=<session_id>
c=<network type> <address type> <connection address>
a=streamControl:<trtsp_url>
a=poseChannel:<trtsp_url> [or] a=poseChannel <multicast_address> <port>
m=<media> [<multicast_port>] <transport> <fmt list>
a=fmtp:<format> <format specific parameters>
g=views <max_views> <min_views> <def_view_group_id1> <def_view_group_id2> ...
g=view <view_group_id> <audioFlag>
c=<network type> <address type> <connection address>
a=streamControl:<trtsp_url>
a=poseChannel:<trtsp_url>
m=<media> [<multicast_port>] <transport> <fmt list>
a=fmtp:<format> <format specific parameters>

```

If all the streams for a session reside on one server,

- i) the connection information is provided once.
- ii) A TRTSP control URL for aggregated control of the entire session is provided
- iii) Optionally, when the client needs to determine the views that are required, a separate pose channel TRTSP URL is set up for the whole session so that a separate SETUP for pose can be negotiated.

If all the view groups have identical formats for the respective media streams, namely video, (audio), and depth map, and all streams exist on a single server, then the m=<media> tags need to be specified only once.

The new g= tag represents the notion of a group. The view group used in telepresence is treated as a specific type of group. The g=views tag is used to specify the maximum number of views in the session (e.g. number of 3DCUs), the minimum number of views needed at a given time in order to render a virtual view (e.g. 2), and the default views with which the session would start. Followed by this, the g=view tag is used to start the description of a particular view group denoted by a unique view\_group\_id within the session. Since audio is optional in this application for a given 3DCU, an audioFlag would indicate the presence or absence of an audio stream associated with that view group. In the event that the view groups are distributed across the network, the connection information for the view group, stream control TRTSP URL, and pose channel TRTSP URL follow this. This is followed by media specific information, where media includes video, audio, and depth maps. In the case of multicast, the port numbers for each media stream is specified here and the connection information for each view group will specify a multicast address with the TTL parameter. The pose channel in the

multicast case will be a separate multicast group, the address of this and the specific port number are specified instead of the TRTSP URL. The type of transport and the format of the media are also specified. Optionally, format specific attributes can be transported as in SDP. Other fields such as time and duration are omitted here for clarity.

Thus, the above TSDP achieves the goals of describing the location of each view group, communicating the media composition of each view group, providing a mechanism for negotiating the transmission of pose information, and providing a mechanism for typical session level stream controls such as play, pause, and seek\_time. The session description is also used by the client to initialize the depacketizers and media decoders.

A sample unicast session description with all the streams on one server will look as follows:

```
s=bike_3d
o=214432432
c=IN IP4 122.133.105.166
a=streamControl:trtsp://dbr.sarnoff.com/bike
a=poseChannel:trtsp://dbr.sarnoff.com/bikepose
m=video RTP/UDP m2v
a=fmtp:m2v main_profile main_level
m=depth RTP/UDP vtx
a=fmtp:vtx version
g=views 8 2 1000 2000
g=view 1000 0
g=view 2000 0
...
g=view 8000 0
```

3. Once a unicast client receives the session description, it will use the TRTSP control channel to specify the client ports for a specific media stream as shown below:

```
SETUP trtsp://tpsurl.com RTSP/1.0
CSeq: 1
Transport: <view_group_id> <media> <transport>;client_port=6666
```

The view\_group\_id and the media uniquely identify the stream for which the client port is specified initially. For example, the client can exchange the port numbers for the streams in the default view groups. As the session evolves and a new view group determined at the client replaces a view group, the client needs to specify the new view\_group\_id that would take the same port number.

4. In response to the SETUP command, the TRTSP channel will specify the server ports for that media stream.

```
CSeq: 1
Session: <session_id>
Transport: <view_group_id> <media> <transport>;client_port=6666;
server_port=5000
```

In the case that an existing port is re-assigned to a new view group, the server would stop the streams on the current `view_group_id`, seek to the same time instant in the new view group streams and use the same server ports used earlier by the old group's streams.

In the case where the server determines that an old view group needs to be replaced by a new view group, it should send a `SETUP` command to the client with the new `view_group_id` associated with the old group's port numbers.

5. A similar `SETUP` command is used to exchange the client and server ports for the pose channel.

Once the client has received the acknowledgements to all the `SETUP` messages, the client can send a `PLAY` command to the server(s). This will initiate the streaming of the pose over the pose channel and the media over the respective RTP channels. The pose messages from the server(s) to the client need to specify the `session_id`, the `view_group_id`, and a time stamp in addition to the pose information. The pose message from the client to the server needs to specify the `session_id` and a time stamp in addition to the pose information. Server and client could negotiate as to where the determination for the required views is done.

In the case of multicast, the session description itself is sufficient for the client to subscribe to the pose channel multicast group and to adaptively subscribe to the desired multicast view groups.

### 5. IMPLEMENTATION DETAILS

Based on the proposed protocol, unicast applications for stored and live streaming were implemented. Figure 2 illustrates the end to end architecture for enabling telepresence. For the stored streaming case,

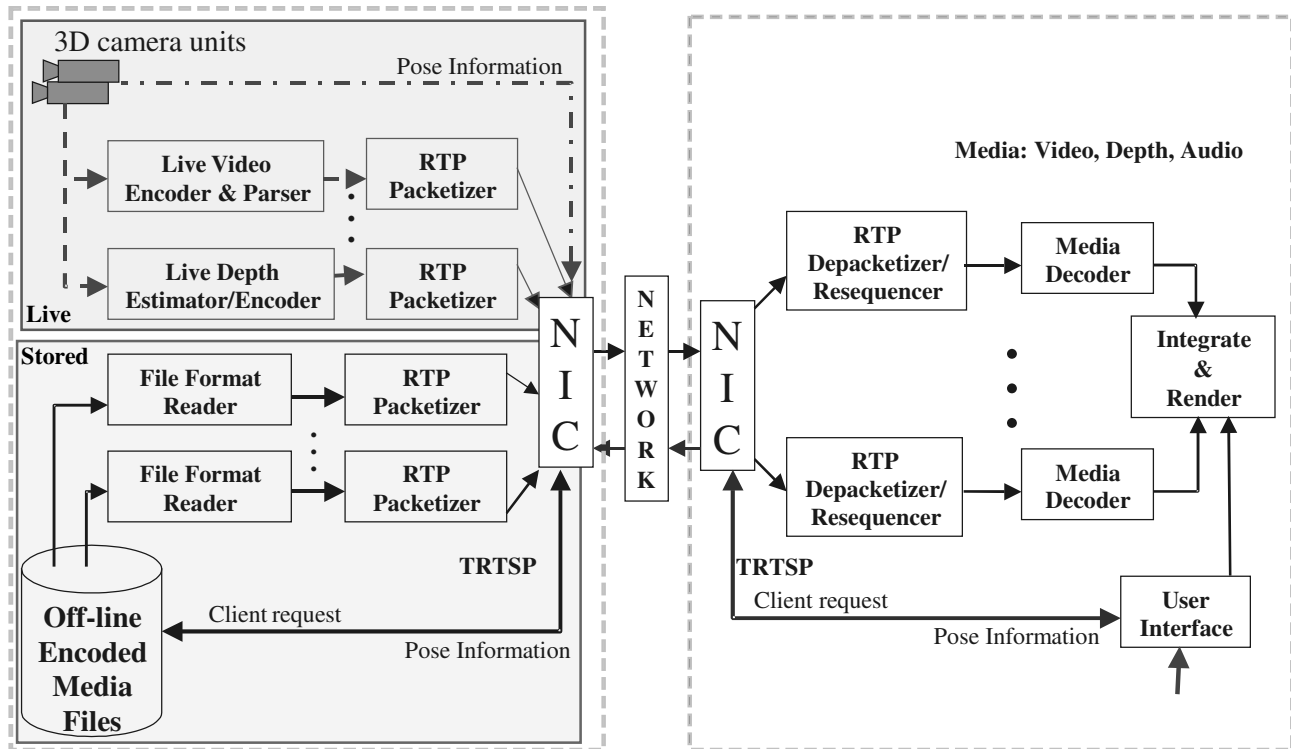
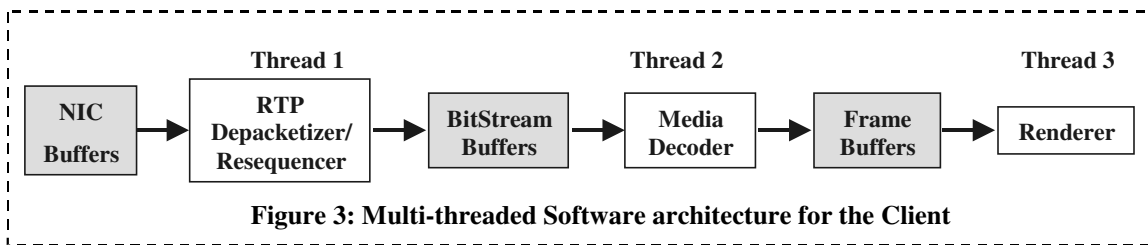


Figure 2: End-to-end streaming architecture for DBR Telepresence

the server employs a configuration file that enables it to open the appropriate media streams and corresponding file formats. The file format for each stream enables the server to randomly access a specific time instant. For the live streaming case, the encoded stream is parsed to extract coding units. Each media stream has different packetization rules depending on the encoding format and the knowledge of independent coding units are important for ensuring error recovery at the client in the event of packet loss. Based on the RTP sequence number and the packetization rules, the depacketizer/resequencer assembles the received packets according to the capabilities of the media decoder. In our experiments, we employed MPEG-2 main profile, main level encoder for video coding and a Sarnoff proprietary format for depth map encoding. The user interface allows the user to navigate in the 3D space and in time. In our experiments, we had only fixed 3DCUs. The client did the determination of which views to request. Only two view groups were used at any time to render a virtual camera view out of a set of 8 camera views at the server. The same RTP ports were re-used when new views were requested.

At the client side, a multi-threaded software architecture was used. A thread-safe shared buffer is used to pass the data between two threads. The threads and buffers used for a single media stream are illustrated in Figure 3. At the server side, an architecture similar to the one described in [4] is used.



The switching delay from one view group to another is not noticeable on a local network. In the stored streaming case, the server is capable of supporting multiple unicast sessions at the same time.

## 6. SUMMARY

Depth-based rendering for telepresence is an interesting application that enables viewers to control their viewpoint in a 3D scene. The aggregation of media streams produced from a single 3D-camera unit gives rise to the concept of a view group. The dynamic pose information of these 3DCUs gives rise to a multitude of possible networking architectures for enabling DBR-telepresence. The most useful of these configurations have been presented in detail. From the concept of view groups and these networking configurations, we have proposed a session protocol that extends RTSP and SDP to meet the specific requirements for the telepresence application. These extensions allow only one control channel to be specified for a view group and enables re-use of ports when switching views in a unicast case. The concept of a group can also be used in other applications (e.g. multiple video objects as in MPEG-4). An end-to-end unicast application has been realized using the proposed protocol. Since a minimum of 2 video streams and 2 depth maps need to be streamed to the client at any given time, the bandwidth requirements are high when compared to last-mile bandwidths available today to consumers. Though we developed this application with the next generation internet in mind, we expect that suitably powerful edge servers (operating in modality 4 discussed in section 2) can make this application available to end customers even in the near term.

## 7. ACKNOWLEDGMENTS

This work was funded by DARPA. I would like to thank Aydin Arpa, Bing-Bing Chai, Paul Hatrack, Ravi Krishnamurthy, and John Wus for helpful discussions on software architecture and for providing feedback by implementing and testing the entire application. I would like to thank Harpreet Sawhney and Rakesh Kumar for discussions and program guidance during the course of the immersive telepresence over the next-generation internet program.

## 8. REFERENCES

- [1] M. Handley, V. Jacobson, SDP: Session Description Protocol, RFC-2327, IETF, April 1998.
- [2] H. Schulzrinne, et. al., Real-Time Streaming Protocol, RFC-2326, April 1998.
- [3] H. Schulzrinne, et. al., RTP: A Transport Protocol for Real-Time Applications, RFC-1889, IETF, January 1996.
- [4] Basso, G. L. Cash, M. R. Civanlar, Real-Time MPEG-2 delivery based on RTP: Implementation Issues, *Signal Processing: Image Communications*, pp. 165-178, vol. 15, 1999.
- [5] Dapeng Wu, et. al., On end-to-end architecture for transporting MPEG-4 video over the Internet, *IEEE Transactions on Circuits and Systems for Video Technology*, Volume: 10 Issue: 6 , Sept. 2000, Page(s): 923 –941.
- [6] M. Levoy and P. Hanrahan. Light field rendering. In *Computer Graphics Proceedings, Annual Conference Series*, pages 31–42, Proc. SIGGRAPH'96 (New Orleans), August 1996. ACM SIGGRAPH.
- [7] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *Computer Graphics Proceedings, Annual Conference Series*, pages 43–54, Proc. SIGGRAPH'96 (New Orleans), August 1996. ACM SIGGRAPH.
- [8] Hai Tao and Harpreet S. Sawhney, Global matching criterion and color segmentation based stereo, in *Proc. Workshop on the Application of Computer Vision (WACV2000)*, pp. 246-253, December 2000.
- [9] Ravi Krishnamurthy, et. al., Compression and transmission of depth maps for image based rendering, *ICIP 2001*, pp. 828-831, October 2001.