# Teleconsultation Enhanced via Session Retrieval Capabilities: Smart Playback Functions and Recovery Mechanism

Pau-Choo Chung and Cheng-Hsiung Wang
*National Cheng Kung University*
*Taiwan*

## 1. Introduction

With the widespread deployment of the Internet nowadays and the increasing power and sophistication of network communication technologies, many collaborative systems have been proposed to support users in geographically dispersed areas in transmitting and sharing multimedia data (Huang et al., 2007; Li et al., 2004; Marsh et al., 2006). One area in which collaborative systems have found particular use is that of telemedicine and teleconsultation, and it is now common practice for physicians to use such systems as a means of analyzing medical images, discussing patients' symptoms, consulting with other medical experts, and so forth (Vazquez et al., 2007; Lee et al., 2004; Lo et al., 2000; Shah et al., 1997; Paul et al., 1998; Guerri et al., 2003; Kholief et al., 2003; Kim et al., 2001). By fully exploiting real-time videoconferencing and medical information sharing, conventional medical teleconsultation systems may satisfy the requirement of providing the interactive discussion environment, but lack session retrieval capabilities that are addressed in terms of session-replay and session-recovery in this chapter.

In medical teleconsultation systems, the ability to replay sessions on demand is of crucial importance since it provides the opportunity to resolve arguments relating to the corresponding case and enables the multimedia content within the session to be used for teaching purposes. However, in the majority of the telemedicine and teleconsultation systems presented in the literature, playback functions are addressed only in passing or have no more than a limited functionality. Shah et al. presented a telemedicine consultation playback system in which a discrete event system specification (DEVS) approach was used to couple the data objects within the system and to model their behavior over time (Shah et al., 1997). However, whilst this approach enabled a synchronization of the various data objects during the playback sequence, the provision of specific playback functions was not considered. In the telemedicine systems (Paul et al., 1998; Guerri et al., 2003), playback functions were provided, but were restricted to chronological order only since all the communication packets within the session were time-stamped to facilitate their synchronization during playback. The event-based and event-tree systems (Kholief et al., 2003; Kim et al., 2001) provide a greater playback flexibility than these time-stamping methods, but lack time-related descriptions and indexes of the objects in the session, and are therefore unable to support playback from randomly specified time points. The ***PlayWatch*** scheme (Tanaka et al., 2005) adopts a chart-style semantic indexing method to locate video scenes and allows users to jump directly to a particular scene simply by selecting an appropriate predefined keyword. However, the system lacks a time index, and thus the process of advancing to a particular time position within video scenes is very slow. The recent MPEG-7 standard (Chang et al., 2006) defines a set of descriptors for describing and indexing video sequences. However, the contents of most video sequences are relatively static, i.e. they do not vary over time. By contrast, medical teleconsultation sessions comprise both image contents and a sequence of commands imposed upon these contents. As a result, the contents of typical medical teleconsultation sequences vary dynamically in accordance with the particular sequence of commands applied to them. Consequently, the scene-based descriptors defined in MPEG-7 are inappropriate for describing and indexing medical teleconsultation sessions.

In developing playback functions for medical teleconsultation sessions, two fundamental issues must be resolved. Firstly, when a playback function is selected, the indexing mechanism of the teleconsultation system must locate the appropriate cut-in point within all the various types of data (e.g. image data, audio data, and so on) which constitute the corresponding scene before the replay process can commence. To reduce the ***restart-latency time*** (i.e. the delay between the moment at which the playback function is invoked and the moment at which playback actually commences), the indexing mechanism must maintain an appropriate cross-linkage amongst the various multimedia data within the session in order to determine the cut-in point in the most efficient manner possible. Secondly, as described above, typical medical teleconsultation sessions involve the use of multiple image processing commands, many of which change the contents of the images within the session. For example, a physician may use a drawing tool to circle a ROI (region-of-interest) on an image and then use a text editing tool to append relevant comments. Furthermore, the modified image contents may be further changed by the invocation of additional commands later in the session. As a result, a strict dependency exists between the image contents and the type and sequential order in which the image processing commands are applied. Consequently, once a suitable cut-in point for a playback function has been located, it is necessary to carry out an appropriate restoration process to restore the image contents from their current condition to that which existed at the cut-in point in the original session.

As mentioned in the above paragraph, the contents of teleconsultation sessions depend critically on the type and sequence of the image processing / analysis commands used during the course of the session. Thus, the reliability of the network connecting the various participants in the session is critical in ensuring that each participant receives a consistent and continuous view of medical images during the on-going session. In practice, however, the network may fail for a variety of hardware or software-related reasons, and thus one or more of the participants are obliged to drop out of the session and re-enter it later. In addition, while some participants take part in a session for its entire duration (e.g. the physician with the overall responsibility for a particular case), others may participate only at a later point in the discussions (e.g. a consultant with input to only one medical image, a physician with the results obtained from medical tests, and so on). For both these “late” users and the “re-entrant” users described above, it is necessary to reconstruct the session contents in such a way that they are able to catch up with the on-going discussions in shortest time as possible.

To support this requirement, some form of “fast forwarding” mechanism is used to advance the re-entrant / late users’ view of the session from its initial state to the current state. This is

commonly achieved by using a centralized content-recording scheme to record the changes in the session contents over the course of the session such that they can be re-executed at the user end as and when required. Existing content-recording schemes can be broadly classified as either checkpoint schemes or message-logging schemes. The former schemes (Gropp & Lusk, 2004; Johnson & Zwaenepoel, 1987; Elnozahy et al., 2002) take periodic snapshots of the session contents and use the latest snapshot as the synchronization point in the restoration process. This method is particularly common in videoconferencing systems in which the session contents at any particular moment in time are independent of the video and audio signals transmitted previously. However, in medical teleconsultation systems, the session contents are critically dependent on the type and sequence of the commands used by the various participants over the course of the session, and thus ensuring the consistency of the session views amongst all the session participants is far more complex. For example, when using the checkpoint scheme to restore the session contents for a re-entrant / late participant, a *rollback propagation* problem (Elnozahy et al., 2002) arises in that it is necessary to rollback the session contents of all the users to the synchronization point. In other words, the restoration process not only causes the on-going session to be suspended, but also loses any updates after the synchronization point. Moreover, the need to back up the entire session contents at each checkpoint incurs a significant overhead. As a result, a tradeoff exists between the frequency at which the checkpoints are refreshed and the performance of the restoration process. Moreover, when the session contents are backed up on a frequency basis in order to ensure the quality of the restoration results, it is possible that in the worst case scenario, the effect of suspending the session for even a very short period of time may result in missing a time-critical aspect of the medical image under discussion.
In contrast to the checkpoint content-recording schemes described above, the message-logging schemes (Elnozahy et al., 2002) sequentially record every message transmitted in a session and reconstruct the session contents from scratch by re-executing each of these messages in the same sequential order. Compared to checkpoint schemes, message-logging schemes not only avoid the worst case scenario described above, but also allow the session to continue without interruption for all the participants other than those for which the restoration process is actually being performed. However, the restoration process is inevitably time consuming since it is necessary to re-execute each and every recorded command for every re-entrant / late user in order to restore the image contents to their current condition. To address this need, a high ***recovery-latency*** delay is induced between the moment at which the re-entrant / late user joins the on-going session and the moment at which the restoration process is completed so that he or she can actively participate in the session. From the above discussions, it is clear that two fundamental issues must be resolved for developing content-recording and restoration schemes for medical teleconsultation sessions. First of all, in restoring the image contents for re-entrant / late users, it is essential that the dependency existing between the image contents and the type and sequential order in which the image processing commands are used must be preserved in order to ensure that each session user has a consistent and up-to-date view of the current session contents. Secondly, the restoration process should be completed as rapidly as possible such that a delay for the re-entrant / late users joining the on-going session can be minimized. Thus, upon implementing restoration schemes for teleconsultation sessions, it is essential to construct an efficient content-recording structure which minimizes the ***recovery-latency*** delay incurred in the restoration process.

Therefore, to design playback functions and recovery mechanism for the teleconsultation sessions, this chapter proposes an enhanced content-recording scheme designated as ***three-level indexing hierarchy*** (***TIH***) which utilizes an efficient cross-linkage design to maintain the dependency between the image contents and the sequential order as the image processing commands are used. As shown in Fig. 1, the ***TIH*** architecture comprises a single *SessionNode* in the first level, a series of *DataNodes* distributed in the time domain in the second level, and a *CommandNode*, *PictureNode* and *UserNode* under each *DataNode* in the third level. Furthermore, ***TIH*** utilizes the command file (Wang et al., 2005) as a ***Command-Record*** file to record all the commands invoked throughout the teleconsultation session, e.g. "mouse move", "mouse click", "draw line", "vertical/horizontal flip", "zoom in/out", "rotation", and so forth. Of these commands, only a limited subset (e.g. "vertical / horizontal flip" and "zoom in / out") actually affect the image appearance. Utilizing a novel cross-linkage design, the ***TIH*** architecture indexes these commands (referred to henceforth as "image-affect commands") such that they can be rapidly identified in the event that a playback function is invoked or a restoration process is required.

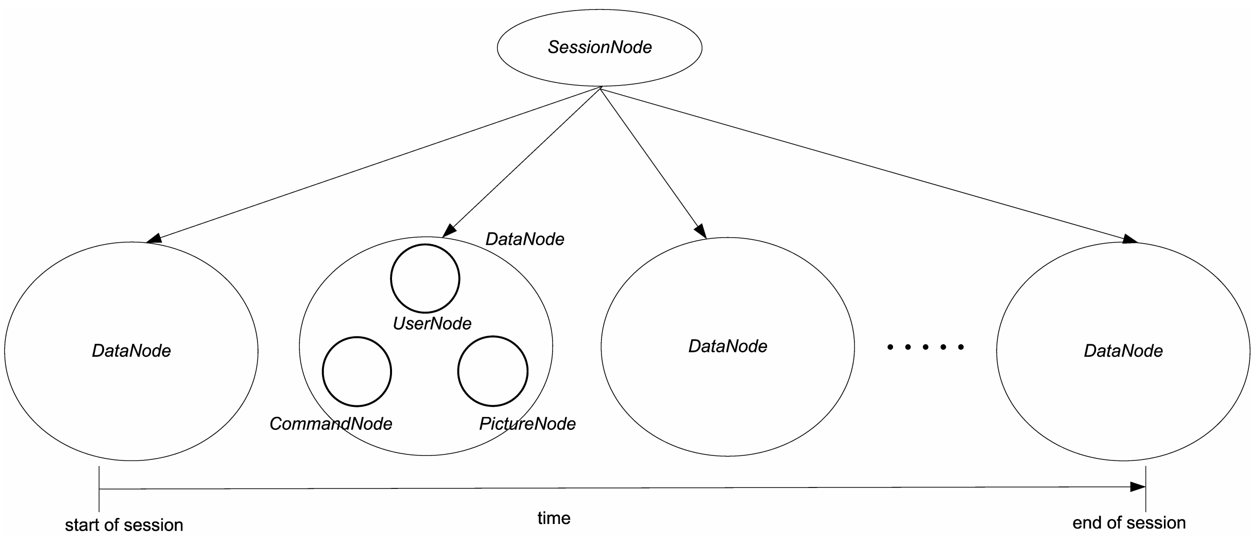


Fig. 1. Illustration of ***TIH*** architecture showing *SessionNode* and *DataNodes* comprising *UserNode*, *CommandNode* and *PictureNode*.

With the help of ***TIH***, four smart playback functions are supported for medical teleconsultation systems, namely (1) replaying the session from a specified point in time, (2) replaying all the segments controlled by a particular physician, (3) replaying all of the segments associated with a specific medical image, and (4) playing a montage of the entire session in order to show its major features. As for the recovery mechanism, the ability of re-entrant / late users to catch up with the on-going discussions in a timely manner can be improved by restoring the foreground image (i.e. the image under current discussion) before the background images (i.e. the remaining images in the session) are restored. Thus, this chapter proposes a novel prioritized recovery policy in which the cross-linkage design of the ***TIH*** architecture is used to accomplish the restoration process free of the dependency constraints between the image contents and the image-affect commands. That is, the foreground image is restored as soon as the re-entrant / late user joins the session and the remaining images in the session are then restored in a transparent background mode. In this way, the user is able to observe the on-going session as the restoration of the background

images proceeds. In this case, the perceived ***recovery-latency*** delay is significantly reduced. When the restoration process is performed, it is possible that the current foreground image may suddenly be replaced by one of the background images. In this event, it is necessary to suspend the current restoration process and to switch to the restoration of the new foreground image. To support this requirement, the proposed prioritized recovery policy maintains a set of resuming pointers for each re-entrant / late user to indicate the current restoration state of each image in the session and facilitate the process of suspending the current restoration process and switching to the restoration of the new foreground image in a timely and computationally-efficient manner.
The remainder of this chapter is organized as follows. Section 2 describes the basic indexing architecture used to facilitate the playback functions in Section 3 and the recovery mechanism in Section 4. Section 3 introduces the cross-linking mechanism used to relate the various data records within the teleconsultation session and explains the role of this mechanism in executing each of the proposed playback functions. Section 4 describes the prioritized recovery policy and discusses the use of the resuming pointers in resolving the foreground image substitution problem during the restoration process. Section 5 quantifies the performance of ***TIH*** in terms of the cut-in point determination time and the image content restoration time for the smart playback functions, and the performance of the proposed recovery mechanism when applied to typical teleconsultation sessions. Finally, Section 6 presents some brief concluding remarks.

## 2. Architecture of *TIH*

Typical medical teleconsultation sessions involve the exchange of multiple media data (e.g. audio, video, medical images, image processing commands, and so forth) between participating physicians. As described in Section 1, a dependency exists between the image contents of such sessions and the sequence and type of the image processing commands invoked during the course of the session. This chapter accounts for this dependency when implementing smart playback functions by utilizing the three-level indexing architecture shown in Fig. 1. Owing to that ***TIH*** is also adopted as the content-recording scheme for the recovery mechanism, the descriptions in this section focused on the smart playback functions are also suitable for the recovery mechanism. In ***TIH***, the *SessionNode* stores the high-level information required to identify the target *DataNode(s)*, while the *DataNodes* contain the detailed information describing all the session events invoked by a particular physician over a specific time period within the session. The details of each of the nodes within ***TIH*** are described in the following paragraphs.

### 2.1 Design of SessionNode

The *SessionNode* has two principal functions, namely to store general information relating to a particular session and to provide basic indexing information such that the target *DataNode(s)* for a particular playback function can be rapidly located. In designing and implementing the *SessionNode*, a number of issues arise. Firstly, it is possible that physicians may download additional medical images on-line during the course of a session to supplement the patient-related images. Irrespective of the playback function invoked by a user, these on-line images should be retrieved only once during the playback procedure in order to minimize the ***restart-latency time***. Thus, in ***TIH***, the information required to retrieve all of the on-line images associated with a particular session is stored in the form of an ***On-line_Index*** in the *SessionNode* such that all of the images can be retrieved in a one-shot process prior to commencing the playback routine. Secondly, a user may request the playback of only those periods of a session for which a certain medical image is discussed. To facilitate this requirement, the *SessionNode* maintains a ***Picture_Index*** to record the *DataNode* – *CommandNode* index pairs for every medical image discussed during the course of the session. Thirdly, a user may only be interested in viewing those parts of a session controlled by a particular physician. Thus, the *SessionNode* maintains a ***User_Index*** to record the indexes of all the *DataNodes* associated with each physician. Finally, the *SessionNode* also records general session information such as the Session ID and the total session time. Fig. 2 illustrates the typical contents of the *SessionNode*.

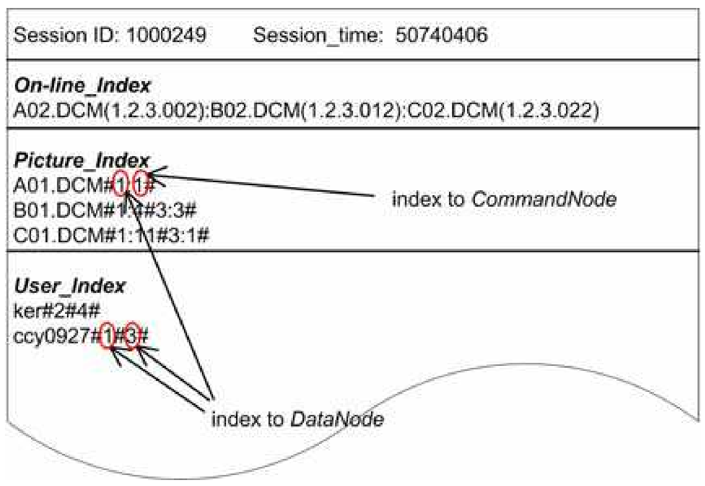


Fig. 2. Typical contents of *SessionNode* in ***TIH*** architecture.

### 2.2 Design of DataNode

The *DataNodes* in ***TIH*** store the index information relating to all the events which take place during the time period(s) within a session for which each particular physician has control. As discussed above, each *DataNode* comprises a *UserNode*, a *CommandNode* and a *PictureNode*. The details of each node are presented in the following.

#### 2.2.1 UserNode

The *UserNode* records the ***Command-Record*** index of the first command performed in the *DataNode*, and maintains information regarding the ***Start_Time*** of the *DataNode* (i.e. the time at which the *DataNode* first became active in the session) and the identity of the physician in control of the *DataNode*. Fig. 3 presents the typical contents of a *UserNode*.

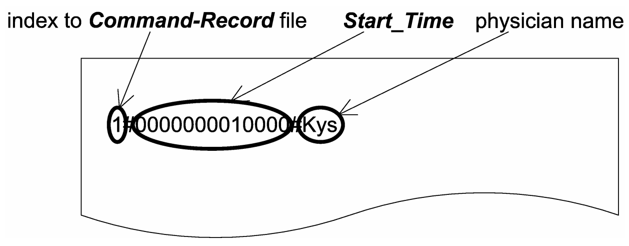
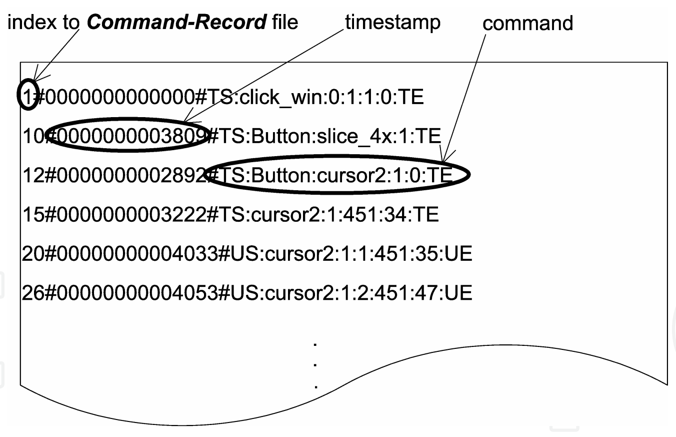


Fig. 3. Typical contents of *UserNode*.

### 2.2.2 CommandNode

The smart playback functions proposed in this chapter may commence at any point within the session. For example, if the user wishes to review the entire session, the cut-in point is located at the very beginning of the session. However, if he or she wishes to view only those segments concerning a particular medical image, the system must jump to the cut-in point corresponding to the first occurrence of this image in the session and then replay the session for so long as the image was discussed. Having done so, it should then jump to the next point in the session at which the image was discussed and replay the corresponding content. This process should be repeated until all of the relevant segments have been located and replayed. Having completed one playback function (e.g. view all segments associated with a particular medical image), the user may decide to replay all of the session segments controlled by a particular physician. In this case, the system is required to "fast forward" or "rewind" from the position at which the previous playback function terminated to the cut-in point associated with the new playback request. However, as described previously, the application of image-affect commands during the course of the session changes the image contents, and thus the current contents (i.e. those associated with the point in the session at which the previous playback function terminated) may well differ from those associated with the new cut-in point. As a result, it is necessary to record all the image-affect commands invoked during the course of the session such that an appropriate subset of these commands can be reapplied to modify the current image contents in such a way as to restore them to their condition at the time point corresponding to the new cut-in point. To support this requirement, all of the image-affect commands performed within each *DataNode* are recorded in the corresponding *CommandNode* together with the index of each command in the ***Command-Record*** file. Fig. 4 shows the typical contents of a *CommandNode*.



```
1#0000000000000#TS:click_win:0:1:1:0:TE
10#0000000003809#TS:Button:slice_4x:1:TE
12#0000000002892#TS:Button:cursor2:1:0:TE
15#0000000003222#TS:cursor2:1:451:34:TE
20#00000000004033#US:cursor2:1:1:451:35:UE
26#00000000004053#US:cursor2:1:2:451:47:UE
.
.
.
```

Fig. 4. Typical contents of *CommandNode*.

### 2.2.3 PictureNode

When a playback function is invoked, it is necessary to determine the particular medical image (defined as the ***target image***) associated with the corresponding cut-in point. In ***TIH***, this requirement is satisfied using the *PictureNode* within each *DataNode*. As shown in Fig. 5, the *PictureNode* contains a series of time-stamped records, each of which represents the introduction of a particular medical image during the course of the session. In addition, the *CommandNode* index of the command used to invoke the image is also recorded.

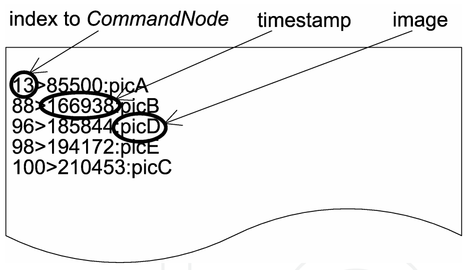


Fig. 5. Typical contents of *PictureNode*.

## 3. Use of *TIH* to facilitate smart playback functions

This section describes the use of the indexing mechanisms described above in locating the cut-in point and identifying the commands required to restore the contents of the target image to an appropriate condition when performing a variety of smart playback functions.

### 3.1 Playback from specified time point in a session

Assume that a particular playback function stops at a certain time point $T_C$ in the session. Assume further that the user subsequently decides to replay the session from another time point, T. Upon receiving this user request, the system performs a binary search on the ***Start_Time*** information recorded at all the *UserNodes* in order to determine the *DataNode* corresponding to time T. Once the target *DataNode* has been located, the system initiates a search for the appropriate cut-in point within this *DataNode*. Fig. 6 presents an illustrative example of the search procedure performed for the case of a playback start time of T=20000. Having identified the target *DataNode*, the target image is determined by searching the corresponding *PictureNode* for the image associated with the timestamp immediately before T, i.e. "picA" in Step (1) of Fig. 6. Using the "index to *CommandNode*" field corresponding to this target image (i.e. "2" in the current example), the *CommandNode* is interrogated to determine the command used to select the target image (i.e. the second record in the

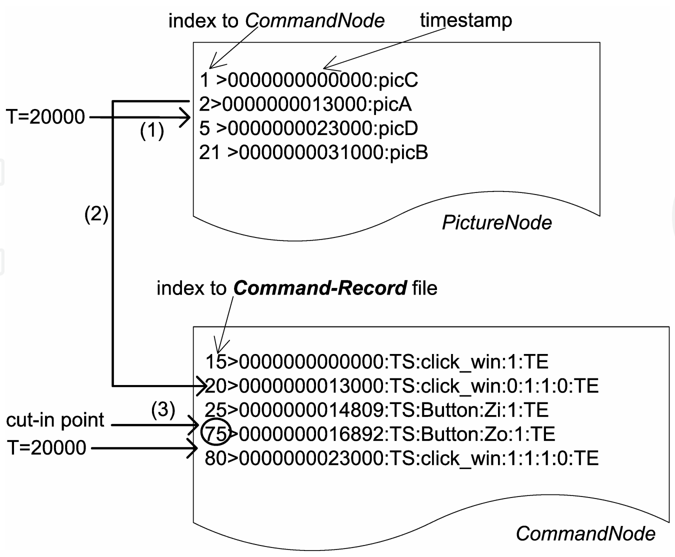

index to CommandNode
timestamp
1 >0000000000000:picC
2>0000000013000:picA
5 >0000000023000:picD
21 >0000000031000:picB
T=20000
(1)
(2)
PictureNode
index to Command-Record file
15>0000000000000:TS:click_win:1:TE
20>0000000013000:TS:click_win:0:1:1:0:TE
25>0000000014809:TS:Button:Zi:1:TE
75>0000000016892:TS:Button:Zo:1:TE
80>0000000023000:TS:click_win:1:1:1:0:TE
cut-in point
(3)
T=20000
CommandNode

Fig. 6. Illustration of cut-in point determination procedure for specified time of T=20000. Note that labels (1), (2) and (3) indicate the three steps performed in locating the cut-in point.

*CommandNode*), as shown in Step (2). Starting from this command, the system then searches the contents of the *CommandNode* to identify the command with the timestamp closest to (but less than) the specified time T (i.e. the fourth record in the *CommandNode*). This command is then identified as the cut-in point for this particular playback request, as shown in Step (3).
Having identified the cut-in point, the actions taken next to restore the image contents depend on the chronological relationship between $T_C$ and T. In the event that $T_C < T$, the playback request prompts a high speed "forward winding" of the session sequence to the cut-in point. Once, it arrives at this point, the system starts to play the session sequence at the normal speed. To accomplish this fast forwarding effect, all of the image-affect commands recorded in the *CommandNode(s)* between $T_C$ and T are sequentially applied to the related medical images in order to restore them to their corresponding states at time T. Conversely, if $T < T_C$, the medical image contents are restored by reapplying all of the image-affect commands invoked between the beginning of the session and time T.

### 3.2 Playback of session segments controlled by specified physician

As described in the following, two different playback functions may be invoked in this particular mode of user request.

#### 3.2.1 Replay from time at which specified physician first takes control of a session

In this scenario, the system locates the first *DataNode* associated with the specified physician by searching the ***User_Index*** in the *SessionNode*. Having done so, the target image is identified from the first record in the *PictureNode* associated with this *DataNode*, while the cut-in point is specified as the first command in the *CommandNode* of the *DataNode*. The restoration mechanism described above is then applied to restore the contents of the medical images to the appropriate condition.

#### 3.2.2 Replay all segments for which specified physician is in control of session

In this scenario, all the *DataNodes* associated with the specified physician are found by searching the ***User_Index*** in the *SessionNode* and these *DataNodes* are then replayed in chronological order. During the replay process, the image contents are restored by sequentially reapplying all the image-affect commands recorded between successive pairs of *DataNodes* during the original session.

### 3.3 Playback of all segments relating to particular medical image

When requested by a user to replay all the session segments relating to a particular medical image, the system interrogates the ***Picture_Index*** in the *SessionNode* of the ***TIH*** architecture to determine all the target *DataNodes* at which the particular medical image is selected and to identify the commands used within these *DataNodes* to select this image. In the illustrative example shown in Fig. 7, the index pair 2:1 indicates that picA was first selected in the second *DataNode* using the first command listed in the corresponding *CommandNode*. Having identified both the *DataNode* and the cut-in point (i.e. the first record in the *CommandNode* of *DataNode* 2), the restoration process described above is performed to restore the target image contents to their original state at the corresponding timestamp. Having performed the playback from the first index pair to the point at which a new image is selected for discussion purposes, the procedure described above is repeated to search for the cut-in point for the next index pair associated with picA.

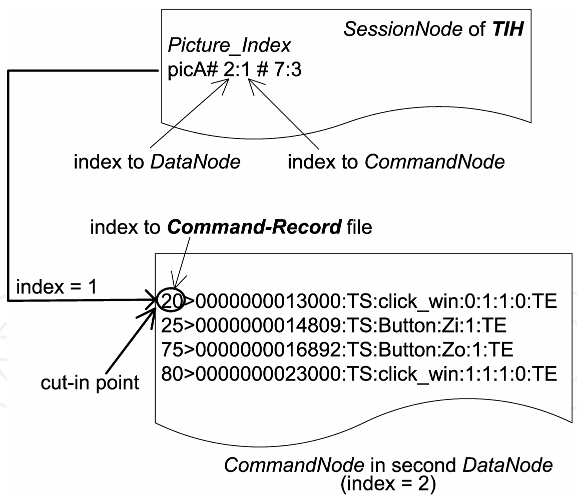

SessionNode of TIH
Picture_Index
picA# 2:1 # 7:3
index to DataNode
index to CommandNode
index to Command-Record file
index = 1
20>0000000013000:TS:click_win:0:1:1:0:TE
25>0000000014809:TS:Button:Zi:1:TE
75>0000000016892:TS:Button:Zo:1:TE
80>0000000023000:TS:click_win:1:1:1:0:TE
cut-in point
CommandNode in second DataNode
(index = 2)

Fig. 7. Illustration of cut-in point determination procedure for specified image picA.

## 3.4 Montage playback - intro-scanning a session

In the movie world, the term "montage" refers to a string of shots extracted from a movie which are spliced together and played contiguously so as to provide the viewer with a brief overview of the entire movie. In compiling a montage of a commercial movie, the director carefully selects certain scenes from the movie to tantalize and intrigue potential moviegoers. By contrast, the montage of a teleconsultation session is generally produced by simply replaying a specified time interval ΔT within every time period T of the session. The parameter ΔT is conventionally referred to as the montage interval and is defined by a starting point and an ending point, respectively. As discussed above, the invocation of image-affect commands has a direct effect on the contents of a medical teleconsultation session. Thus, in the montage playback function proposed in this chapter, the occurrence of any image-affect command during the time period T is automatically taken as the starting point for the following montage interval ΔT (see Fig. 8) such that the montage contains all the most significant scenes within the session. Note that in the event that no image-affect commands are invoked during a particular time period T, the montage playback function simply selects a general command once the current time period expires. In order to implement the montage playback function, the system records the starting and ending points of each montage interval ΔT associated with a particular session in a ***Montage-Record*** file with the structure shown in Fig. 9. Note that in this figure, the M_point field contains

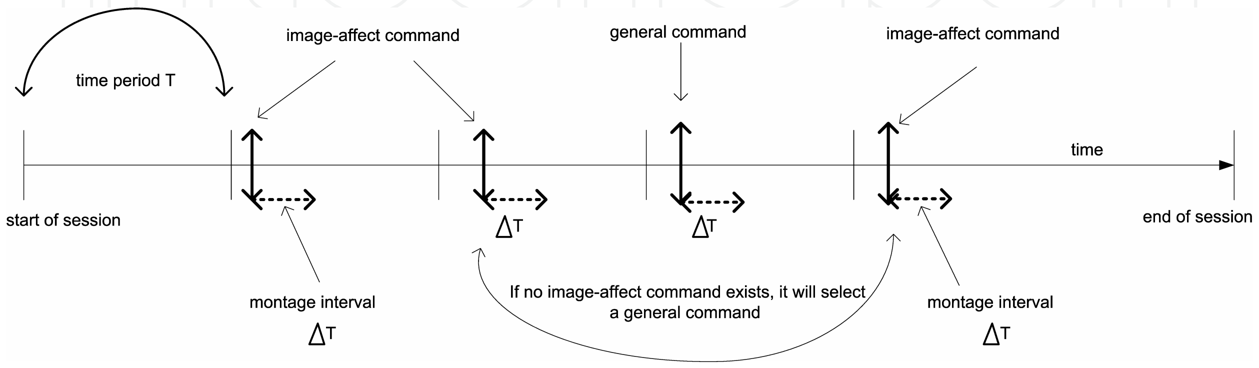


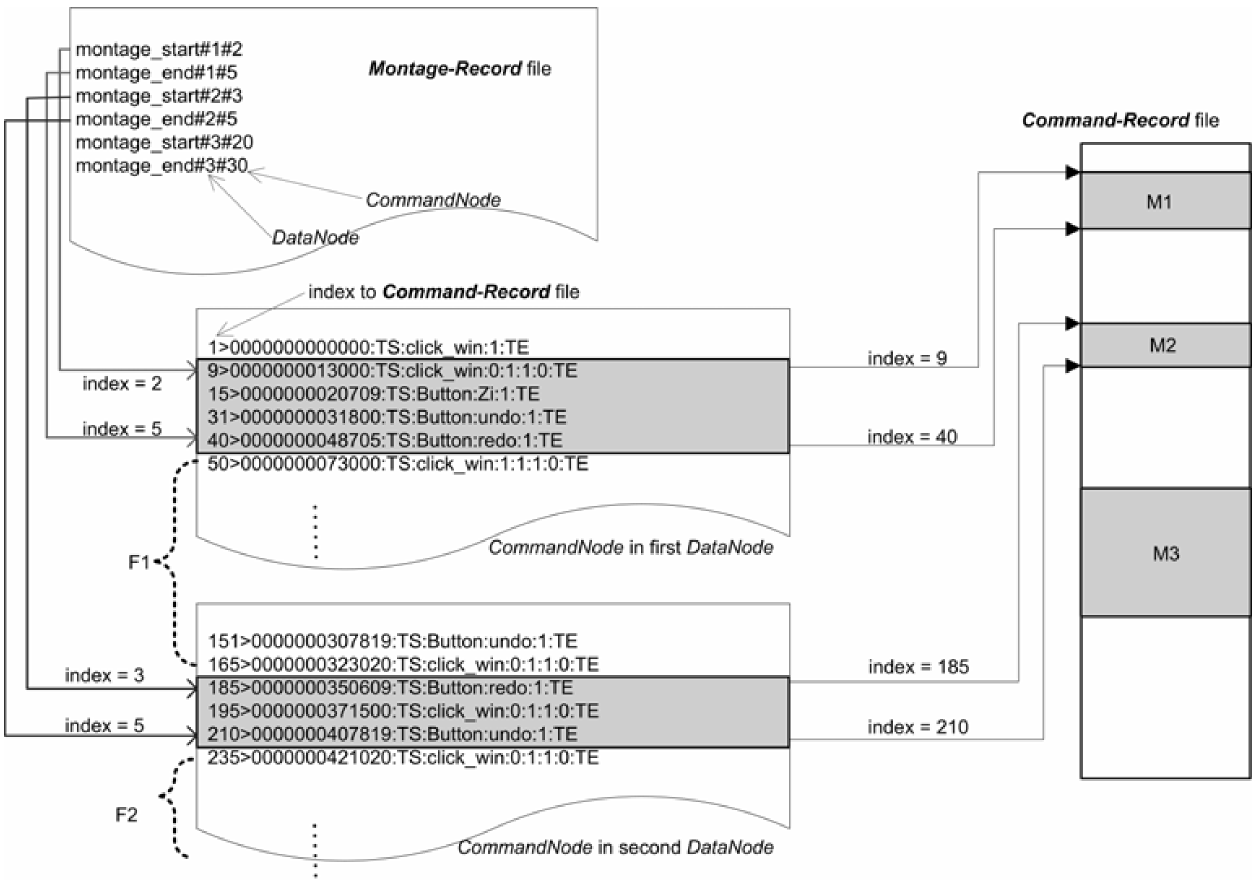
Fig. 8. Selection of montage playback points in each montage interval.

| | | | | |
|---|---|---|---|---|
| M_point | # | D_Node | # | Cmd_Node |

Fig. 9. Structure of records in ***Montage-Record*** file.

either "montage_start" (indicating the starting point of the montage interval) or "montage_end" (indicating the ending point). In addition, the D_Node field specifies the *DataNode* containing each montage interval, while the Cmd_Node field indicates the index of the *CommandNode* associated with the corresponding starting or ending point. Finally, the hash symbol (#) simply denotes a field separator.

Fig. 10 presents an illustrative example of the ***Montage-Record*** file and the indexing procedure applied when executing the montage playback function. The first and second records in the ***Montage-Record*** file represent the starting and ending points of the first montage interval (M1), respectively. Similarly, the third and fourth records define the second montage interval (M2), the fifth and sixth records define the third montage interval (M3), and so on. The first record in the ***Montage-Record*** file, i.e. "montage_start#1#2", indicates that this particular montage interval starts from the second record listed in the *CommandNode* associated with the first *DataNode*. By examining the "index to ***Command-Record*** file" field associated with this record (value = 9), the system determines the starting



```
montage_start#1#2
montage_end#1#5
montage_start#2#3
montage_end#2#5
montage_start#3#20
montage_end#3#30
```

```
1>0000000000000:TS:click_win:1:TE
9>0000000013000:TS:click_win:0:1:1:0:TE
15>0000000020709:TS:Button:Zi:1:TE
31>0000000031800:TS:Button:undo:1:TE
40>0000000048705:TS:Button:redo:1:TE
50>0000000073000:TS:click_win:1:1:1:0:TE
```

```
151>0000000307819:TS:Button:undo:1:TE
165>0000000323020:TS:click_win:0:1:1:0:TE
185>0000000350609:TS:Button:redo:1:TE
195>0000000371500:TS:click_win:0:1:1:0:TE
210>0000000407819:TS:Button:undo:1:TE
235>0000000421020:TS:click_win:0:1:1:0:TE
```

Fig. 10. Illustrative example of ***Montage-Record*** file showing determination of montage intervals M1, M2 and M3 in ***Command-Record*** file and restoration intervals F1 and F2 in *CommandNodes* of corresponding *DataNodes*.

point of the first montage interval in the ***Command-Record*** file. Meanwhile, the second record in the ***Montage-Record*** file, i.e. "montage_end#1#5", indicates that this particular montage interval ends with the fifth record listed in the *CommandNode* associated with the first *DataNode*. By examining the entry in the "index to ***Command-Record*** file" field associated with the fifth record in the *CommandNode* (value = 40), the system locates the ending point of the montage interval in the ***Command-Record*** file. Thus, the shaded area labeled M1 in the ***Command-Record*** file contains all the commands applied to the image contents during this particular montage interval. By repeating this process for all the montage intervals within the session, the system compiles all the commands applied during the intervals of interest in the session (indicated by the shaded areas M1, M2 and M3 in Fig. 10). During the montage playback procedure, the commands listed in the intervals F1 and F2 of the *CommandNode(s)* are used to fast forward the session sequence from the end of one montage interval to the beginning of the next, while the commands within the shaded areas of the ***Command-Record*** file in Fig. 10 are applied to the image contents at a normal speed in order to replay the session in its original form within each montage interval.

## 4. Use of *TIH* to facilitate recovery mechanism

As discussed in Section 1, it is necessary to perform the image content restoration process for two particular classes of participants, namely late and re-entrant participants. From the perspective of the session server, the mechanisms used to restore the session contents are identical for both types of participants. In both cases, the overriding requirement is to maintain the dependency between the image contents and the sequential order in which the image processing commands were used. As will be described in the following sub-sections, the cross-linkage design of the ***TIH*** architecture not only satisfies this dependency requirement, but also facilitates a prioritized recovery policy which enables re-entrant / late users to catch up with the on-going session with a minimal possible ***recovery-latency*** delay.

### 4.1 Prioritized recovery policy

In traditional message-logging based restoration systems, the server transmits every recorded command to the re-entrant / late user, and these commands are then re-executed in order to replicate the actions applied to the session contents before he / she joins the session. Whilst this approach results in an accurate reconstruction of the session contents, sequentially executing all of the commands transmitted from the session server is highly time consuming and forces the participant to wait before the participant can take an active part in the on-going discussions. Accordingly, the proposed ***TIH*** architecture is designed to accelerate the restoration process by using two key strategies, namely a selective transmission of the logged commands to the re-entrant / late user and a prioritized recovery policy.

As described in Section 2, all the commands invoked by each user during the session are recorded in the ***Command-Record*** file (Wang et al., 2005). However, of these commands, only those which change the image contents in some way are indexed at the *CommandNode* in each *DataNode*. In the restoration process, the session server transmits only the image-affect commands stored in the *CommandNodes* to re-entrant / late users. As a result, a significant reduction in the ***recovery-latency*** delay is gained compared to that used in a traditional message-logging scheme. The perceived ***recovery-latency*** delay is further reduced by using a prioritized recovery policy to preferentially restore the foreground image (i.e. the image under current discussion) before any of the background images (i.e. the other images in the session) is restored. In other words, the image-affect commands related to the background images are not transmitted to the re-entrant / late user until the foreground image has been restored. As a result, the user is able to observe (but not participate in) the on-going session as the remaining images are restored transparently in the background.

The restoration process of a medical image in a session can be explained by means of the pseudo codes in Fig. 11. In Step 1 of Fig. 11, the name of the image that is to be restored is input. In Step 2, the session server initializes the restoration process by interrogating the ***Picture_Index*** in the *SessionNode* of ***TIH*** to retrieve the number of *DataNode* – *CommandNode* index pair associated with the specific image (designated as $\boldsymbol{n}$). Also, the serial number of the *DataNode* – *CommandNode* index pair being processed (designated as $\boldsymbol{i}$) is set to 0, representing the first index pair. In Step 3, in the event that $\boldsymbol{i} < \boldsymbol{n}$, Step 4 is performed; otherwise, Step 8 is performed. In Step 4, the *DataNode* at which the specific image was selected for discussion and the command used within the *DataNode* to select this image will be identified (the indexing process of Step 4 will be illustrated in the next paragraph). In Step 5, the *DataNode* and the command used to select another image for discussion will be identified (the indexing process of Step 5 will be illustrated in the next paragraph). In Step 6, the commands starting from the command found in Step 4 and ending with the one immediately prior to the command found in Step 5 will be re-executed, regarded as the restoration process of the $\boldsymbol{i}$th *DataNode* – *CommandNode* index pair. In Step 7, the value of $\boldsymbol{i}$ is added with 1 and Step 3 is invoked again. Step 8 of Fig. 11 indicates that the whole restoration process for the specific image is completed.

```
Step 1: INPUT the name of the image to be restored.
Step 2: GET the number of DataNode – CommandNode index pair associated with the image (designated as $n$);
        SET the serial number of the DataNode – CommandNode index pair being processed (designated as $i$) to 0.
Step 3: if $i < n$ then GOTO Step 4; otherwise, GOTO Step 8.
Step 4: INDEX the command used to select the image for discussion.
Step 5: INDEX the command used to select another image for discussion.
Step 6: RUN the commands starting from the command found in Step 4 and ending with the one immediately prior
        to the command found in Step 5.
Step 7: ADD 1 to $i$; GOTO Step 3.
Step 8: END
```

Fig. 11. The pseudo codes for the restoration process of certain medical image in a session.

To explain the indexing process in Steps 4 and 5 of Fig. 11, Fig. 7 presents an illustrative example in which picA is assumed to be restored for a re-entrant / late user entering the on-going session. The *DataNode* – *CommandNode* index pair 2:1 in the ***Picture_Index*** indicates that picA was selected for discussion in the second *DataNode* using the first command listed in the corresponding *CommandNode* (The command "TS:click_win" indicates the selection of a new image). Similarly, the index pair 7:3 shows that picA was later selected as the foreground image in the seventh *DataNode* using the third command in the corresponding *CommandNode*. Having identified all the *DataNodes* and commands used to select picA for discussion, the session server selects all the commands in the corresponding *CommandNodes* starting from the command used to select the image and ending with the command immediately prior to that used to select a new image as the foreground image, and then transmits these commands to the re-entrant / late user end, where they are re-executed. For example, taking the first index pair in Fig. 7 as an example, the restoration process commences with the first command in the *CommandNode* at *DataNode 2* and terminates after the third command (since the fourth command, TS:click_win, is used to select a new image for discussion purposes). Having performed the restoration of the first index pair, the procedure is repeated for all the other index pairs associated with picA.

### 4.2 Resolving suspension / resumption issues when performing restoration process

Since the *CommandNode* indexes only those commands which directly affect the session contents, a parsing of the entire ***Command-Record*** file is not required. As a result, the time required to restore the foreground image is considerably shorter than that required when a traditional message-logging based system is used. Furthermore, since the restoration process commences with the foreground image, the participant is able to follow the on-going session as the remaining images in the session are restored in a transparent background mode. However, it is possible that another medical image may be selected as a new foreground image while the restoration process is still on going. In this situation, no matter whether the session server is currently performing the restoration of the foreground image or background images, it must suspend its restoration activities and resume (or start) the restoration of this new foreground image. To facilitate this suspension / resumption process, the session server creates a set of resuming pointers for each re-entrant / late user to indicate the current restoration state of each image in the session. In implementing this approach, a resuming pointer is appended to the next image-affect command associated with the restored image as soon as the previous image-affect command associated with the same image has been transmitted by the session server to the re-entrant / late user. If when processing the image-affect commands in the *CommandNode*, it is found that the resuming pointer points at the final image-affect command relating to the current image, the restoration of the image is completed as soon as this image-affect command has been transmitted to the user. Thus, the resuming pointer is removed from the set of resuming pointers to indicate that this particular image is now fully restored for this particular user. In addition, if the removed resuming pointer is associated with the foreground image, i.e. the foreground image has been fully restored, all of the image-affect and non-image-affect commands subsequently applied to this image by the other participants in the session are transmitted to the re-entrant / late user such that he / she can follow the on-going discussions in a passive mode as the background images are restored.
Upon completing the restoration of the foreground image the session server randomly picks a new resuming pointer from the set of resuming pointers and performs the restoration of the corresponding background image. The restoration process for a particular user is considered to be complete when the set of resuming pointers associated with that user is empty. Until then the re-entrant / late user can apply commands to the session contents in the same way as any other participant in the session. Note that this restriction is deliberately imposed in order to prevent re-entrant / late users from inadvertently selecting an image currently under restoration as the new foreground image, thereby resulting in an inconsistency between their versions of the image contents and that of the remaining participants in the session.

## 5. Evaluation results

### 5.1 Performance of *TIH* in facilitating smart playback functions

In this section, the feasibility of ***TIH*** was explored by performing a series of playback experiments using representative medical teleconsultation sessions. Fig. 12 presents a snapshot of the operational interface for the particular case in which the user requests the system to replay specific or all the session segments associated with a particular physician. In evaluating the performance of any indexing mechanism designed for playback purposes, the most important parameters include the ***target-setting time*** (i.e. the time spent by the user interacting with the system in specifying the target scene to be replayed) and the ***restart-latency time*** (i.e. the elapsed time between the moment at which the playback function is invoked and that at which the playback actually commences). The ***target-setting time*** of the ***TIH***-based system proposed in this chapter was evaluated using the same experimental method as that used for ***PlayWatch*** (Tanaka et al., 2005). Specifically, eight medical staff with different levels of computer literacy were requested to perform four different playback tasks using the proposed playback system, namely: (1) playback from a specified time point in a session, (2) playback of all the session segments controlled by a specified physician, (3) playback of all the segments relating to a particular medical image, and (4) montage playback. The time spent by each user in specifying the target scene for each playback task was measured and taken as the corresponding ***target-setting time***. Meanwhile, the time between the moment at which the user invoked the playback function and the moment at which playback actually commenced was measured and taken as the ***restart-latency time***. In ***TIH***, the ***restart-latency time*** has two components, namely the ***search time***, i.e. the time required to locate the cut-in point, and the ***restoration time***, i.e. the time required to restore the image contents. Therefore, in quantifying the performance of the playback scheme, the search time and the restoration time were individually recorded for each of the playback tasks assigned to the eight users. The ***restart-latency time*** varies directly with the extent to which image-affect commands are invoked during the session. Thus, to evaluate the performance of the ***TIH***-based playback scheme under realistic conditions, three different session types, each with a different number of image-affect commands per minute, were designed, namely Type A with around 5 image-affect commands per minute, Type B with around 10 image-affect commands per minute, and Type C with around 20 image-affect commands per minute. Furthermore, to reflect the differing durations of typical real-world medical teleconsultation sessions, each type of session was run for both 10 and 30 minutes, respectively.

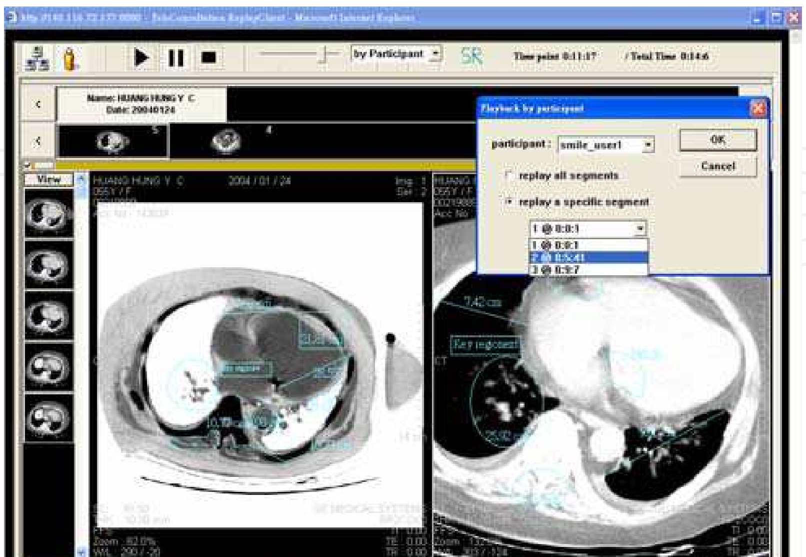


Fig. 12. Snapshot of operational interface for case where playback system is set to replay session segments associated with specified physician.

Fig. 13 and Fig. 14 present the evaluation results obtained for the ***target-setting time*** and the ***restart-latency time*** for each of the eight users when applying the playback functions to the 10-minute and 30-minute sessions, respectively. Note that all of the experiments were performed on a PC with a P4 2.4 GHz CPU and 1 GB of RAM. In both figures, it can be seen that the ***target-setting time*** varies notably from one user to the next due to their differing levels of computer literacy. However, no more than a slight variation is observed in the ***restart-latency time*** for each user. Tables 1 and 2 present a detailed breakdown of the ***restart-latency time*** for the 10-minute and 30-minute sessions, respectively. It can be seen that the search time required to locate the cut-in point varies in the range 7.4 ~ 17.4 ms for the three session types and two session durations considered in the evaluation trials. Comparing the search times associated with the different playback functions, it is found that playback from a specified time point in the session (designated as "Time" in Tables 1 and 2) induces the longest search time (i.e. 15.6 ~ 17.4ms), while the montage playback (Montage) induces the shortest search time (i.e. 7.4 ~ 9.6 ms). These findings are reasonable since playback from a specified time in the session contents requires the use of a binary search to locate the corresponding target *DataNode*, while in the montage playback function, the cut-in points are already pre-designated as the starting points of the montage intervals in the ***Montage-Record*** file. Furthermore, as shown in Tables 1 and 2, the search time accounts for only a very small proportion of the total ***restart-latency time***. In other words, the ***restart-latency time*** is dominated by the restoration time in every case. As implied above, it is reasonable to assume that the restoration time varies linearly with the number of image-affect commands applied during the restoration process. Fig. 15 confirms that this is indeed the case for both the 10-minute and 30-minute sessions. Tables 1 and 2 also reveal that for each session duration (i.e. 10 minutes or 30 minutes), the restoration time increases significantly as the density of the image-affect commands increases. Similarly, for each type of session (i.e. Type A, Type B or Type C), the restoration time increases dramatically as the length of the session is increased. By contrast, it can be seen that the search time component of the ***restart-latency time*** not only has a relatively small value compared to the restoration time (i.e. a mean value of just 11.8 ~ 12.65 ms), but also remains approximately constant for each of the different session durations and session types. In other words, the efficiency of the proposed ***TIH*** architecture in locating the cut-in point is confirmed.

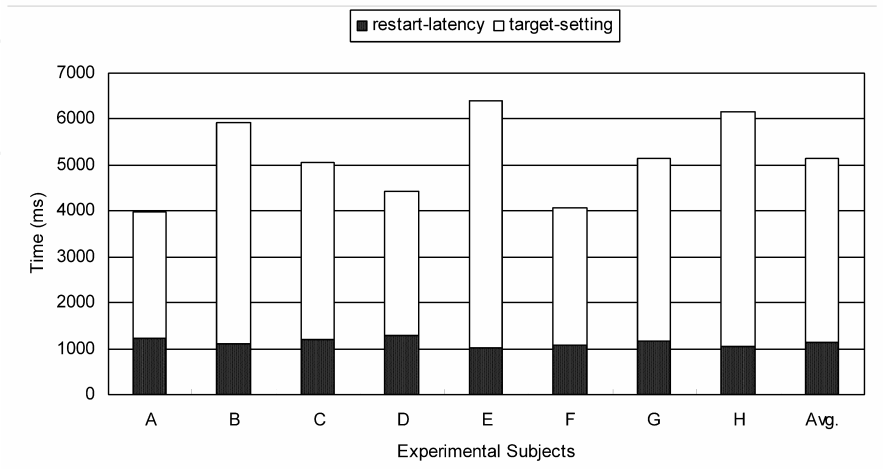


Fig. 13. Evaluation results for ***target-setting time*** and ***restart-latency time*** in 10-minute sessions.

In a recent study, a "scene-retrieval" system designated as ***PlayWatch*** was proposed, in which scene descriptors derived from the MPEG-7 standard were used to facilitate playback functions similar to those proposed in this chapter. However, the indexing mechanism proposed in this chapter is far more efficient than ***PlayWatch*** in terms of its storage capacity requirements. For example, in the proposed system, a typical 60-minute session generates a 6 Mbyte audio file, a 2 Mbyte ***Command-record*** file, and a 0.5 Mbyte ***TIH*** architecture. By contrast, ***PlayWatch*** requires a total of 88.9 Mbytes of storage capacity (with a CBR (compression bit rate) of 127.8 Kbytes/s) to store the scenes within a 12-minute video sequence. Moreover, ***PlayWatch*** requires an average of 20 seconds to retrieve a specified scene and spends around 50 seconds in locating a specific time position within the video sequence (Tanaka et al., 2005). By contrast, ***TIH*** has an average ***target-setting time*** of 4 seconds (see Figures 13 and 14), and most importantly, incurs an average ***restart-latency time*** of just 1 second for 10-minute sessions and 3 seconds for 30-minute sessions (see Tables

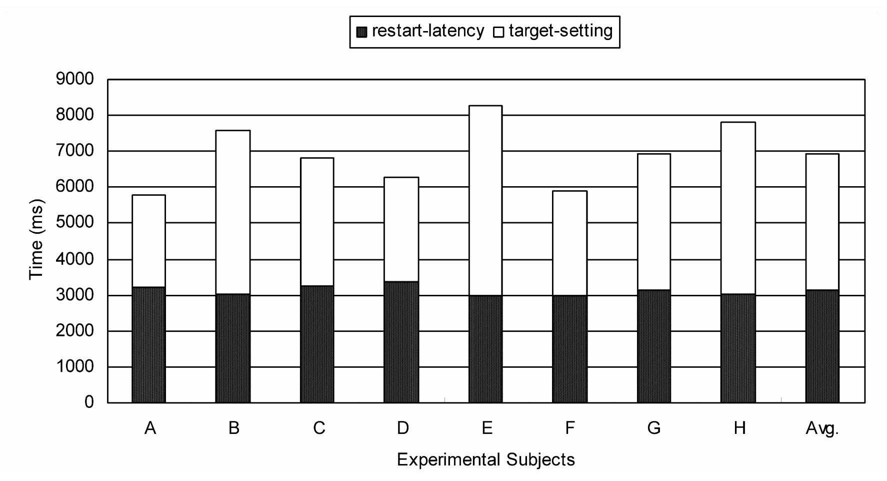


Fig. 14. Evaluation results for ***target-setting time*** and ***restart-latency time*** in 30-minute sessions.

| Session type \ Playback functions \ Measure item | | Search time (ms) | Restoration time (ms) | Number of restored image-affect commands | Restart-latency time (ms) |
|---|---|---|---|---|---|
| Type A | Time | 15.9 | 387 | 27 | 402.9 |
| | Physician | 12.4 | 248 | 17 | 260.4 |
| | Image | 12.6 | 385 | 25 | 397.6 |
| | Montage | 8.7 | 368 | 20 | 376.7 |
| | Average | 12.4 | 347 | 22.25 | 359.4 |
| Type B | Time | 16.5 | 1035 | 67 | 1051.5 |
| | Physician | 12.8 | 1012 | 65 | 1024.8 |
| | Image | 12.2 | 1092 | 74 | 1104.2 |
| | Montage | 8.3 | 1020 | 60 | 1028.3 |
| | Average | 12.45 | 1039.75 | 66.5 | 1052.2 |
| Type C | Time | 16.2 | 2219 | 158 | 2235.2 |
| | Physician | 12.2 | 1425 | 98 | 1437.2 |
| | Image | 12.6 | 2422 | 166 | 2434.6 |
| | Montage | 9.6 | 1912 | 128 | 1921.6 |
| | Average | 12.65 | 1994.5 | 137.5 | 2007.15 |

Table 1. Performance evaluation results for playback functions in 10-minute sessions.

| Session type | Playback functions | Search time (ms) | Restoration time (ms) | Number of restored image-affect commands | Restart-latency time (ms) |
|---|---|---|---|---|---|
| Type A | Time | 16.9 | 1195 | 81 | 1211.9 |
| | Physician | 13.2 | 1294 | 88 | 1307.2 |
| | Image | 12.2 | 1365 | 88 | 1377.2 |
| | Montage | 8.1 | 1570 | 98 | 1578.1 |
| | Average | 12.6 | 1356 | 88.75 | 1368.6 |
| Type B | Time | 15.6 | 2055 | 142 | 2070.6 |
| | Physician | 12.6 | 2797 | 188 | 2809.6 |
| | Image | 11.6 | 2130 | 145 | 2141.6 |
| | Montage | 7.4 | 2702 | 174 | 2709.4 |
| | Average | 11.8 | 2421 | 162.25 | 2432.8 |
| Type C | Time | 17.4 | 5054 | 323 | 5071.4 |
| | Physician | 12.6 | 4895 | 314 | 4907.6 |
| | Image | 12.5 | 6705 | 392 | 6717.5 |
| | Montage | 7.5 | 5580 | 345 | 5587.5 |
| | Average | 12.5 | 5558.5 | 343.5 | 5571 |

Table 2. Performance evaluation results for playback functions in 30-minute sessions.

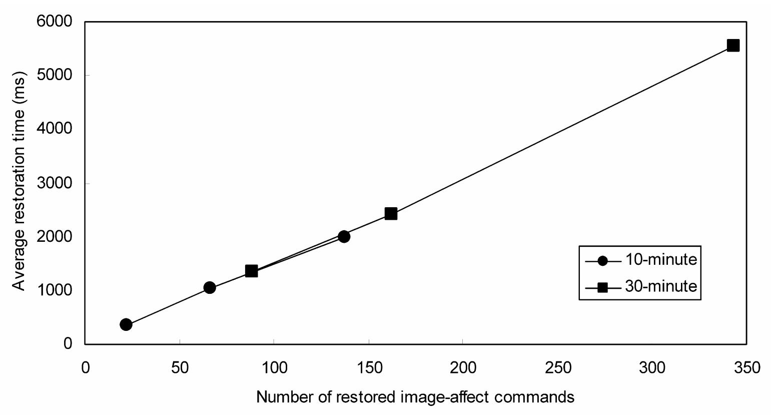


Fig. 15. Variation of restoration time with number of restored image-affect commands for sessions of two different durations.

1 and 2). In other words, the proposed ***TIH*** architecture yields a significant improvement in the cross-linking efficiency of the playback system, and therefore reduces the search time considerably compared to that of ***PlayWatch***.

### 5.2 Evaluation of proposed recovery mechanism

The performance of the proposed recovery mechanism was evaluated by performing a series of experiments in which disconnection failures were mimicked by intentionally unplugging the network connections of certain participants in a medical teleconsultation session and measuring the resulting ***recovery-latency***. Fig. 16 illustrates the experimental environment consisting of three clients and a server connected through the Internet. Here, Client_A is a PC with an Intel Pentium 4 2.4 GHz processor and 1 GB RAM; Client_B is a laptop with an Intel Pentium M 1.5 GHz processor and 768 MB RAM; Client_C is a PC with an Intel Pentium 4 2.8 GHz processor and 1 GB RAM; and the Session_Server is a PC with two Intel Xeon 2.4 GHz processors and 2 GB RAM. In evaluating the performance of any recovery mechanism (a conventional checkpoint or message-logging system or that proposed in this chapter), one of the most important performance parameters is that of the ***recovery-latency***, i.e. the elapsed time between the moment at which the re-entrant / late user joins the on-going session and the moment at which the restoration process is completed and the user can participate in the on-going discussions in a normal manner. In order to demonstrate the efficiency of the recovery mechanism proposed in this chapter, the ***recovery-latency*** is measured for two different recovery policies, namely a basic recovery policy and the prioritized recovery policy. The basic recovery policy simply re-executes every command invoked since the session begins, whereas the prioritized recovery policy uses ***TIH*** to identify and reapply only those commands which directly affect the image contents. In addition, the performance of the prioritized recovery policy is further evaluated by measuring the ***foreground sync-time***, i.e. the time for which the re-entrant / late user can watch the on-going session but can not take an active part in the discussions since the restoration of the background images is not yet fully completed.

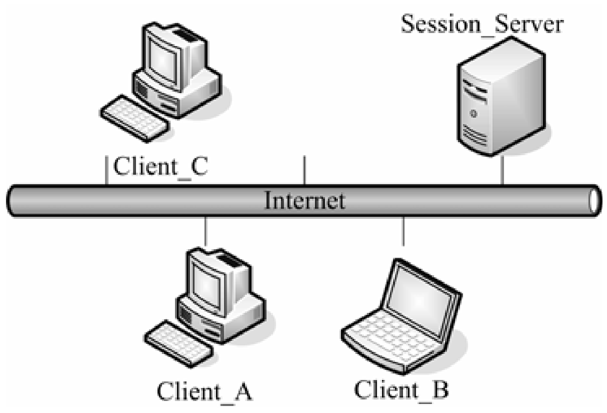


Fig. 16. Experimental environment.

In performing the evaluation experiments, the supported image-affect commands included the following: “black and white inversion”, “90 degree rotation”, “180 degree rotation”, “270 degree rotation”, “vertical flip”, “horizontal flip”, “zoom in”, and “zoom out”. The session included a total of 10 medical images, comprised 10 non-image-affect commands (i.e. “mouse move”) per second, involved a total of 160 image-affect commands, and was run for 20 minutes. The probability density function describing the application of each image-affect command to each image was modeled by a normal distribution with mean of ($\mu$) = 2 and a standard deviation of ($\sigma$) = 1. A total of five experimental patterns were obtained for evaluation purposes, as listed in Tables 3 to 7.
According to a series of experiments, the execution time of each image-affect command was evaluated at Client_A and Client_B, respectively. Table 8 presents the corresponding results obtained when averaging the execution time over 100 separate measurements of each image-affect command. Table 9 summarizes the ***recovery-latency*** at Client_A and Client_B for each of the five experimental patterns under the basic and prioritized recovery policies. It is observed that the average ***recovery-latency*** at Client_A was reduced from 58553 ms to 13206 ms when the basic recovery policy was replaced with the prioritized recovery policy. In other words, the ***recovery-latency*** at Client_A was reduced by around 77.45% when the proposed recovery mechanism was used. Similarly, the average ***recovery-latency*** at Client_B was reduced from 62909 ms of using the basic recovery policy to 15974 ms of using the

| Image number | Black and white inversion | 90° rotation | 180° rotation | 270° rotation | Vertical flip | Horizontal flip | Zoom in | Zoom out |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 0 | 2 | 2 | 4 | 3 | 1 | 2 |
| 2 | 1 | 2 | 1 | 3 | 2 | 0 | 2 | 3 |
| 3 | 3 | 2 | 3 | 2 | 2 | 2 | 1 | 1 |
| 4 | 2 | 2 | 2 | 1 | 2 | 3 | 3 | 2 |
| 5 | 1 | 3 | 2 | 0 | 3 | 4 | 3 | 0 |
| 6 | 2 | 4 | 2 | 2 | 0 | 2 | 2 | 3 |
| 7 | 0 | 2 | 2 | 3 | 2 | 1 | 2 | 2 |
| 8 | 2 | 1 | 4 | 1 | 2 | 2 | 1 | 4 |
| 9 | 4 | 2 | 2 | 2 | 1 | 2 | 3 | 1 |
| 10 | 3 | 2 | 0 | 4 | 2 | 1 | 2 | 2 |

Table 3. Contents of experimental pattern 1

| Image number | Black and white inversion | 90° rotation | 180° rotation | 270° rotation | Vertical flip | Horizontal flip | Zoom in | Zoom out |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 1 | 2 | 2 | 4 | 2 | 2 |
| 2 | 2 | 3 | 2 | 1 | 1 | 2 | 3 | 2 |
| 3 | 3 | 2 | 3 | 2 | 2 | 0 | 1 | 3 |
| 4 | 2 | 1 | 2 | 3 | 2 | 1 | 2 | 3 |
| 5 | 1 | 2 | 2 | 1 | 3 | 3 | 2 | 2 |
| 6 | 2 | 3 | 3 | 0 | 1 | 2 | 2 | 2 |
| 7 | 2 | 4 | 3 | 2 | 2 | 1 | 2 | 0 |
| 8 | 3 | 1 | 2 | 3 | 2 | 2 | 3 | 1 |
| 9 | 2 | 2 | 0 | 2 | 2 | 2 | 2 | 3 |
| 10 | 2 | 0 | 2 | 4 | 3 | 3 | 1 | 2 |

Table 4. Contents of experimental pattern 2

| Image number | Black and white inversion | 90° rotation | 180° rotation | 270° rotation | Vertical flip | Horizontal flip | Zoom in | Zoom out |
|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 1 | 2 | 2 | 3 | 1 | 2 | 3 |
| 2 | 1 | 2 | 2 | 3 | 2 | 2 | 3 | 1 |
| 3 | 2 | 3 | 3 | 2 | 1 | 1 | 1 | 2 |
| 4 | 2 | 2 | 1 | 0 | 2 | 4 | 2 | 4 |
| 5 | 2 | 3 | 2 | 3 | 3 | 2 | 1 | 0 |
| 6 | 1 | 3 | 2 | 2 | 1 | 3 | 2 | 1 |
| 7 | 2 | 1 | 2 | 1 | 2 | 2 | 3 | 2 |
| 8 | 2 | 2 | 4 | 2 | 2 | 0 | 2 | 3 |
| 9 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 10 | 3 | 1 | 0 | 3 | 2 | 3 | 2 | 2 |

Table 5. Contents of experimental pattern 3

| Image number | Black and white inversion | 90° rotation | 180° rotation | 270° rotation | Vertical flip | Horizontal flip | Zoom in | Zoom out |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 2 | 2 | 3 | 2 | 2 |
| 2 | 2 | 4 | 1 | 2 | 3 | 2 | 1 | 2 |
| 3 | 3 | 2 | 2 | 1 | 2 | 1 | 2 | 3 |
| 4 | 2 | 0 | 2 | 3 | 1 | 2 | 4 | 1 |
| 5 | 1 | 2 | 3 | 2 | 2 | 2 | 2 | 2 |
| 6 | 2 | 2 | 1 | 1 | 2 | 4 | 2 | 2 |
| 7 | 3 | 1 | 2 | 2 | 1 | 2 | 3 | 1 |
| 8 | 2 | 2 | 4 | 2 | 2 | 1 | 2 | 2 |
| 9 | 1 | 3 | 2 | 3 | 2 | 2 | 0 | 3 |
| 10 | 2 | 3 | 0 | 2 | 3 | 1 | 2 | 2 |

Table 6. Contents of experimental pattern 4

prioritized recovery policy, corresponding to a performance improvement of 74.61%. The experimental results in Table 9 also shed light on the relative effects of the "image-affect" and "non-image-affect" commands on the performance of the restoration process. In the prioritized recovery policy, only the "image-affect" commands were reapplied during the restoration process, and thus the measured value of the ***recovery-latency*** provided an indication of the weight of the "image-affect" commands in the restoration process. By contrast, in the basic recovery policy, both the "image-affect" and the "non-image-affect" commands were re-executed during the restoration process. Consequently, the difference in the ***recovery-latency*** times of the basic and prioritized recovery policies provided an indication of the effect of the "non-image-affect" commands on the performance of the restoration process. In Table 9, when the additional weight of the "non-image-affect" commands was ignored, the ***TIH*** / prioritized recovery policy yielded a significant improvement in the restoration process. In practice, this performance improvement can be attributed to two main factors, namely (1) ***TIH*** facilitates the rapid identification of all the image-affect commands applied to the image of interest when performing the restoration process; and (2) the server transmits only those commands which directly affect the image contents, thereby reducing both the transmission time and the restoration time.

| Image number | Black and white inversion | 90° rotation | 180° rotation | 270° rotation | Vertical flip | Horizontal flip | Zoom in | Zoom out |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 3 | 2 | 3 | 2 | 1 | 2 |
| 2 | 1 | 2 | 2 | 3 | 2 | 2 | 3 | 1 |
| 3 | 1 | 3 | 2 | 1 | 2 | 3 | 3 | 2 |
| 4 | 2 | 2 | 4 | 2 | 3 | 1 | 1 | 2 |
| 5 | 2 | 2 | 0 | 2 | 1 | 2 | 3 | 3 |
| 6 | 2 | 2 | 1 | 4 | 3 | 0 | 0 | 3 |
| 7 | 3 | 1 | 2 | 1 | 2 | 3 | 2 | 2 |
| 8 | 2 | 3 | 2 | 3 | 1 | 2 | 2 | 1 |
| 9 | 2 | 1 | 2 | 2 | 2 | 3 | 1 | 2 |
| 10 | 3 | 3 | 2 | 0 | 1 | 2 | 4 | 2 |

Table 7. Contents of experimental pattern 5

| | Black and white inversion | 90° rotation | 180° rotation | 270° rotation | Vertical flip | Horizontal flip | Zoom in | Zoom out |
|---|---|---|---|---|---|---|---|---|
| Client_A | 46.27 | 84.28 | 87.35 | 86.24 | 80.25 | 82.72 | 50.24 | 51.64 |
| Client_B | 56.25 | 106.45 | 109.65 | 107.15 | 97.75 | 96.24 | 58.85 | 59.55 |

Table 8. Average execution time for each image-affect command (unit: ms)

| | Recovery policy | Pattern 1 | Pattern 2 | Pattern 3 | Pattern 4 | Pattern 5 | Average | Improvement |
|---|---|---|---|---|---|---|---|---|
| Client_A | Prioritized | 13117 | 13275 | 13463 | 12980 | 13195 | 13206 | 77.45% |
| | Basic | 58625 | 58005 | 58755 | 59125 | 58255 | 58553 | |
| Client_B | Prioritized | 15817 | 16173 | 16281 | 15735 | 15864 | 15974 | 74.61% |
| | Basic | 62755 | 63125 | 63015 | 62886 | 62764 | 62909 | |

Table 9. Experimental results: ***recovery-latency*** (unit: ms)

Table 10 summarizes the ***recovery-latency*** and ***foreground sync-time*** at Client_A and Client_B for each of the five experimental patterns when the prioritized recovery policy was implemented. It can be seen that the ***foreground sync-time*** is around 10.9~11.2 % (i.e. 1/10th) of the ***recovery-latency***. That is, the restoration time of the foreground image was just 1/N of the total ***recovery-latency*** time, where N is the number of medical images in the session. In other words, the prioritized recovery policy enabled the re-entrant / late users to join the on-going session in a passive capacity within a very short period of time following their entry to the session. As a result, the users perceived a significant reduction in the ***recovery-latency*** delay compared to that in a traditional message-logging based scheme.

| | Time | Pattern 1 | Pattern 2 | Pattern 3 | Pattern 4 | Pattern 5 | Average | Percentage |
|---|---|---|---|---|---|---|---|---|
| Client_A | ***foreground sync-time*** | 1436 | 1452 | 1458 | 1412 | 1442 | 1440 | 10.9% |
| | ***recovery-latency*** | 13117 | 13275 | 13463 | 12980 | 13195 | 13206 | |
| Client_B | ***foreground sync-time*** | 1780 | 1794 | 1809 | 1771 | 1786 | 1788 | 11.2% |
| | ***recovery-latency*** | 15817 | 16173 | 16281 | 15735 | 15864 | 15974 | |

Table 10. Experimental results: ***foreground sync-time*** and ***recovery-latency*** (unit: ms)

In the following we evaluated the performance of the proposed recovery mechanism at Client_A in Fig. 16 for sessions characterized by different numbers of medical images, non-image-affect commands, and image-affect commands, respectively, and different session times. To evaluate the effect of the number of medical images on the ***recovery-latency*** and ***foreground sync-time***, three different sessions were designed, namely Session#1 with 5 medical images, Session#2 with 10 medical images, and Session#3 with 20 medical images. Note that the other experimental conditions for each session (i.e. the number of non-image-affect commands, the number of image-affect commands, and the session time) were assigned the same values as those used for the initial series of experiments. The corresponding results in Fig. 17 show that the ***recovery-latency*** remained approximately constant as the number of images in the session increased. However, the ***foreground sync-time*** decreased approximately linearly as the number of images is increased. In other words, an increasing number of images had no significant effect on the restoration process, but yielded a noticeable reduction in the ***foreground sync-time***.

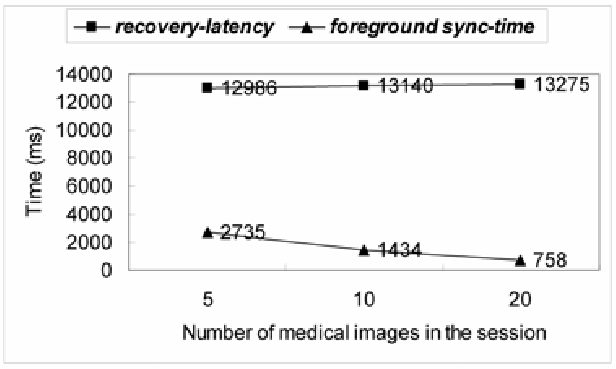


Fig. 17. Experimental results for ***recovery-latency*** and ***foreground sync-time*** in sessions with different numbers of medical images.

To evaluate the effect of the number of non-image-affect commands on the performance of the proposed recovery mechanism, three experimental sessions were designed with different "mouse move" command rates, i.e. 5 times per second (Session#1), 10 times per second (Session#2) and 20 times per second (Session#3). Note that the other experimental conditions (i.e. the number of medical images, the number of image-affect commands, and the session time) were specified in accordance with the values assigned in the original experiments. The experimental results are presented in Fig. 18. It was seen that both the ***recovery-latency*** and the ***foreground sync-time*** were insensitive to the number of non-image-affect commands.

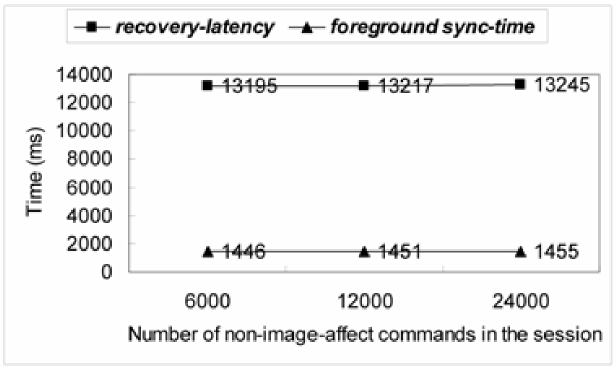


Fig. 18. Experimental results for ***recovery-latency*** and ***foreground sync-time*** in sessions with different numbers of non-image-affect commands.

The influence of the number of image-affect commands on the performance of the proposed recovery mechanism was evaluated using three different sessions, namely Session#1 with 80 image-affect commands, Session#2 with 160 image-affect commands, and Session#3 with 320 image-affect commands. Note that in each session, the image-affect commands were averagely distributed over the 10 medical images. The results presented in Fig. 19 show that the ***recovery-latency*** and the ***foreground sync-time*** both increased linearly with the number of image-affect commands. However, it can be seen that the number of image-affect commands had a greater effect on the ***recovery-latency*** time than on the ***foreground sync-time***. Since both the ***recovery-latency*** and the ***foreground sync-time*** were induced as a result of the re-execution of image-affect commands in the restoration process, the finding in Fig. 19 is expected since the ***recovery-latency*** and ***foreground sync-time*** both increased with an increasing number of image-affect commands. As discussed previously, the restoration of the foreground image accounted for approximately 1/N of the total restoration time, where N represents the number of medical images in the session and is equal to 10 in Fig. 19. Therefore, the rate at which the ***foreground sync-time*** increased with an increasing number of image-affect commands is around 1/10th that of the increase in the ***recovery-latency*** time. For example, when the number of image-affect commands was increased from 160 to 320, Fig. 19 shows that the ***recovery-latency*** increased by about 12640 ms while the ***foreground sync-time*** increased by about 1313 ms.

Finally, the effect of the session duration on the performance of the proposed recovery mechanism was evaluated using three sessions of varying lengths, i.e. 10 minutes (Session#1), 20 minutes (Session#2) and 40 minutes (Session#3). The corresponding results are shown in Fig. 20. It can be seen that neither the ***recovery-latency*** nor the ***foreground sync-time*** was significantly affected by the session time. In general, the results presented in Fig. 17 ~ 20 show that the performance of the proposed recovery mechanism was insensitive to the number of medical images, the number of non-image-affect commands, and the session time. The ***recovery-latency*** inevitably increased with the number of image-affect commands. Nonetheless, the proposed prioritized recovery policy minimized the effect of an increasing number of image-affect commands on the ***foreground sync-time***, and thus

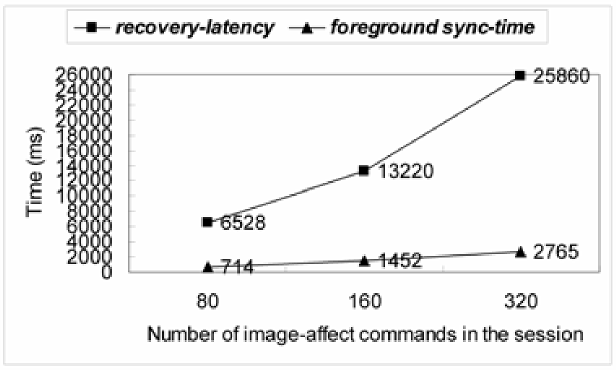


Fig. 19. Experimental results for ***recovery-latency*** and ***foreground sync-time*** in sessions with different numbers of image-affect commands.

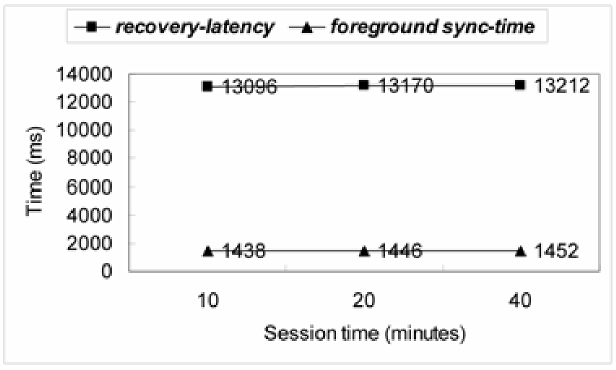


Fig. 20. Experimental results for sessions with different session times.

re-entrant / late users were able to rapidly join the session (albeit in a passive manner) even in sessions characterized by a large number of such commands.

## 6. Conclusion

This chapter has presented an indexing scheme designated as ***three-level indexing hierarchy*** (***TIH***) to support a range of smart playback functions and a novel recovery mechanism for teleconsultation sessions. Uniquely, the contents of such sessions are command dependent, i.e. the contents vary in accordance with the commands applied to them during the course of the session. Furthermore, when executing smart playback functions, the playback sequences invariably commence at different points within the session. As a result, it is necessary to "fast forward" or "fast rewind" the session contents to an appropriate cut-in point and to restore the contents to the corresponding condition before the playback procedure can commence. In this chapter, the content restoration process is achieved by reapplying an appropriate sub-set of the image-affect commands applied to the original image contents during the actual session. The efficiencies of the cut-in point determination process and the image content restoration procedure, respectively, are enhanced via the cross-linkage structure of ***TIH*** which records the time-based changes in the various types of multimedia content within the session and maintains a link between these changes and the commands which caused them. ***TIH*** makes possible a range of smart playback functions, including replaying from a specified time point within the session, replaying all the segments of a session controlled by a particular physician, replaying all the session segments for which a particular medical image is discussed, and playing a montage of the entire session. The evaluation results have shown that the proposed indexing mechanism yields a significant improvement in both the ***restart-latency time*** and the storage capacity requirements of the playback system compared to existing scene-based playback systems such as ***PlayWatch***. As for the recovery mechanism, a prioritized recovery policy is proposed to accomplish the preferential restoration of the foreground image (i.e. the medical image under current discussion) prior to the background images (i.e. all the other images in the session). The prioritized recovery policy enables re-entrant / late users to follow the on-going session in a passive capacity as the restoration of the remaining images is completed in a transparent background mode. Upon implementing the prioritized recovery policy for each re-entrant / late user, the suspension / resumption problem which arises when the foreground image is suddenly replaced by a new image before the restoration process is fully completed is managed by utilizing a set of resuming pointers to indicate the current restoration state of each medical image. The evaluation results have confirmed that the ***TIH*** / prioritized recovery policy yields a significant improvement in the ***recovery-latency*** delay compared to the traditional message-logging restoration scheme. In addition, the results have shown that the prioritized recovery policy enables re-entrant / late users to participate passively in the on-going session within 1/N of the total ***recovery-latency*** delay time, where N is the number of medical images in the session. Finally, it has been shown that the ***recovery-latency*** of the proposed recovery mechanism is insensitive to the number of medical images in the session, the number of non-image-affect commands, and the session time, respectively.

## 7. References

S.F. Chang; T. Sikora & A. Puri (2006). Overview of the MPEG-7 standard, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 11, No. 6, June 2006, pp. 688-695.

E. N. Elnozahy; L. Alvisi; Y. M. Wang & D. B. Johnson (2002). A survey of rollback-recovery protocols in message-passing systems, *ACM Computing Surveys*, Vol. 34, No. 3, Sept. 2002, pp. 375-408.

William Gropp & Ewing Lusk (2004). Fault Tolerance in Message Passing Interface Programs, *The International Journal of High Performance Computing Applications*, Vol. 18, No. 3, Fall 2004, pp. 363-372.

J.C. Guerri; C.E. Palau; A. Pajares; A. Belda; J.J. Cermeno & M. Esteve (2003). A Multimedia Telemedicine System to Assess Musculoskeletal Disorders, *Proceedings of the 2003 International Conference on Multimedia and Expo*, Vol. 1, July 2003, pp. 1-701-4.

W. Huang; Y. Ai; Z. Chen; Q. Wu; H. Ouyang; P. Jiao; Z. Liu & C. Fang (2007). Computer Supported Cooperative Work (CSCW) for Telemedicine, *Computer Supported Cooperative Work in Design, 2007. CSCWD 2007. 11th International Conference on*, April 2007, pp. 1063-1065.

D. B. Johnson & W. Zwaenepoel (1987). Sender-based message logging, *17th international symposium on fault tolerant computing*, July 1987, pp. 14-18.

Kholief M.; Maly K. & Shen S. (2003). Event-Based Retrieval from a Digital Library Containing Medical Streams, *Proceedings of the 2003 Joint Conference on Digital Libraries*, May 2003, pp. 231-233.

Yeongho Kim; Jeong-Ho Choi; Jongki Lee; Myeng Ki Kim; Nam Kuk Kim; Jin Sup Yeom & Yong Oock Kim (2001). Collaborative surgical simulation over the Internet, *IEEE Internet Computing*, Vol. 5, No. 3, May-June 2001, pp. 65-73.

Chien-Cheng Lee; Pau-Choo Chung; Yunghsiang Sam Han; Dyi-Rong Duh & C. W. Lin (2004). A Practice of a Collaborative Multipoint Medical Teleconsultation System on Broadband Network, *Journal of High Speed Networks*, Vol. 13, No. 3, 2004, pp. 207-222.

J.J. Li; T. Li; Z. Lin; A. Mathur & K. Kanoun (2004). Computer supported cooperative work in software engineering, *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, Sept. 2004, pp. 328.

Chien-Shun Lo; Ching-Wen Yang; Pau-Choo Chung; Yen-Chien Ouyang; San-Kan Lee & Ping-Song Liao (2000). A Mammography Tele-Consultation Pilot System in Taiwan, *Journal of High Speed Networks*, Vol. 9, 2000, pp. 31-46.

Marsh J.; Glencross M.; Pettifer S. & Hubbold R. (2006). A network architecture supporting consistent rich behavior in collaborative interactive applications, *Visualization and Computer Graphics, IEEE Transactions on*, Vol. 12, No. 3, May-June 2006, pp. 405-416.

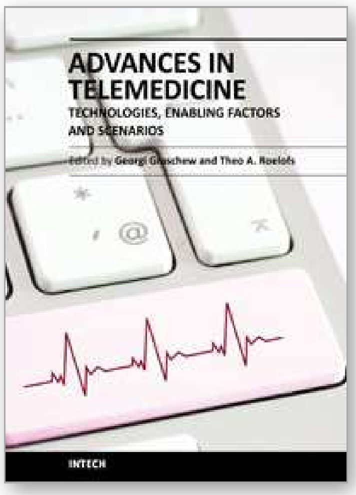
Paul B.B. & Civanlar M.R. (1998). VTJukebox: implementation issues for RTP-based recording and on-demand multicast of multimedia conferences, *Multimedia Signal Processing, 1998 IEEE Second Workshop*, Dec. 1998, pp. 259-264.

P.J. Shah; R. Martinez & B.P. Zeigler (1997). Design, Analysis, and Implementation of a Telemedicine Remote Consultation and Diagnosis Session Playback Using Discrete Event System Specification, *IEEE Trans. on Information Technology in Biomedicine*, Vol. 1, No. 3, Sept. 1997, pp. 179-188.

K. Tanaka; T. Sasaki; Y. Tonomura; T. Nakanishi & N. Babaguchi (2005). PlayWatch: Chart-Style Video Playback Interface, in *Proc. ICME 2005*, 2005, pp. 731-734.

Jose Carlos Dafonte Vazquez; Alfonso Castro Martinez; Angel Gomez & Bernardino Arcay Varela (2007). Intelligent agents technology applied to tasks scheduling and communications management in a critical care telemonitoring system, *Computers in Biology and Medicine*, Vol. 37, No. 6, June 2007, pp. 760-773.

C.H. Wang; C.C. Lee; H.C. Jiau; P.C. Chung; T.L. Yang; K.F. Ssu & Y.S. Kuo (2005). Improving the Tele-consultation Services – Capabilities of Retro, in *Proceedings of the 3rd European Medical & Biological Engineering Conference*, 2005.



**Advances in Telemedicine: Technologies, Enabling Factors and Scenarios**
Edited by Prof. Georgi Graschew

ISBN 978-953-307-159-6
Hard cover, 412 pages
**Publisher** InTech
**Published online** 16, March, 2011
**Published in print edition** March, 2011

Innovative developments in information and communication technologies (ICT) irrevocably change our lives and enable new possibilities for society. Telemedicine, which can be defined as novel ICT-enabled medical services that help to overcome classical barriers in space and time, definitely profits from this trend. Through Telemedicine patients can access medical expertise that may not be available at the patient's site. Telemedicine services can range from simply sending a fax message to a colleague to the use of broadband networks with multimodal video- and data streaming for second opinioning as well as medical telepresence. Telemedicine is more and more evolving into a multidisciplinary approach. This book project "Advances in Telemedicine" has been conceived to reflect this broad view and therefore has been split into two volumes, each covering specific themes: Volume 1: Technologies, Enabling Factors and Scenarios; Volume 2: Applications in Various Medical Disciplines and Geographical Regions. The current Volume 1 is structured into the following thematic sections: Fundamental Technologies; Applied Technologies; Enabling Factors; Scenarios.

**How to reference**
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Pau-Choo Chung and Cheng-Hsiung Wang (2011). Teleconsultation Enhanced via Session Retrieval Capabilities: Smart Playback Functions and Recovery Mechanism, Advances in Telemedicine: Technologies, Enabling Factors and Scenarios, Prof. Georgi Graschew (Ed.), ISBN: 978-953-307-159-6, InTech, Available from: http://www.intechopen.com/books/advances-in-telemedicine-technologies-enabling-factors-and-scenarios/teleconsultation-enhanced-via-session-retrieval-capabilities-smart-playback-functions-and-recovery-m