



RACOL

Rural Advanced Community of Learners

Capture - Delivery System

The fifth item of *Project Deliverables*, Schedule A of the RACOL project reads.

- **A Protocol for “time stamping” and storing the video, audio and computer data so that a learning session can be reproduced at a later time.**

This “Protocol” as built consists of a mechanism to collect audio, video and computer data which is stored for later use, and a mechanism to allow the students to conveniently review the data offline.

This document describes the *capture system* and the *delivery system*.

Design goals at the onset were:

- The preferred equipment was to be oriented around the Windows operating system
- The system was to operate without operator intervention for daily operation.
- Existing facilities and software were to be used. Special purpose solutions were to be minimized.
- Internal to the Ft Vermillion school district bandwidth requirements were to be kept to 250 kilobits per second.
- A modem level stream was to be produced.
- The capture system was to be controlled exclusively by the “Scheduler” which also controls the VLPE.
- Access to the collected material was to be restricted to students in the Ft. Vermillion system.
- The captured material was to be available to students during the process of capturing it.

The summary section at the end of this document assess the success and failures in meeting these goals.

1.1 Capture System

The capture system is a self contained unit which can sit anywhere in the FVSD network. It requires physical proximity to the MCU for Audio and Video feed and for communications from the Scheduler.

The hardware is an Apple XServe with Audio/Video interface boxes connected directly to the MCU Audio/Video output lines. The Apple XServe is on loan from Apple Canada until September 2004.

The Serial control line provided by the Scheduler initiates and terminates recording of ongoing classroom activity. No other control facility exists.

Storage of the captured material is on the XServe computer.

1.2 Delivery System

The delivery system consists of a Web Server providing student access to the collected material, and a Video Streaming Server that replays the collected sessions across the network.

The Web Server and the Streaming Server are hosted on a Windows 2000 Server provided by FVSD.

1.3 Capture System Overview

The drawing in Figure 1 shows the connections between the Multipoint Control Unit and the Capture System. In the existing configuration there are two computers required for this configuration. Collection is performed by the Apple XServe. The Windows Capture Station is required to provide an image of the smart board activity for the collection process.

In the Figure the box Control Process is shown as being connected to the scheduler. The Control process starts, stops and monitors the collection process and formats and stores the collected material in a form that the *Delivery System* can use.

The interface devices between the Multipoint Control Unit and the Capture system are required for encoding of the Audio and Video signals.

The video encoding hardware used is the Imaging Source Video to Firewire converter (DFG/1394-1). This device converts an input analog video stream to an uncompressed IEEE-1394 video stream. Further details on this device may be obtained from www.theimagingsource.com.

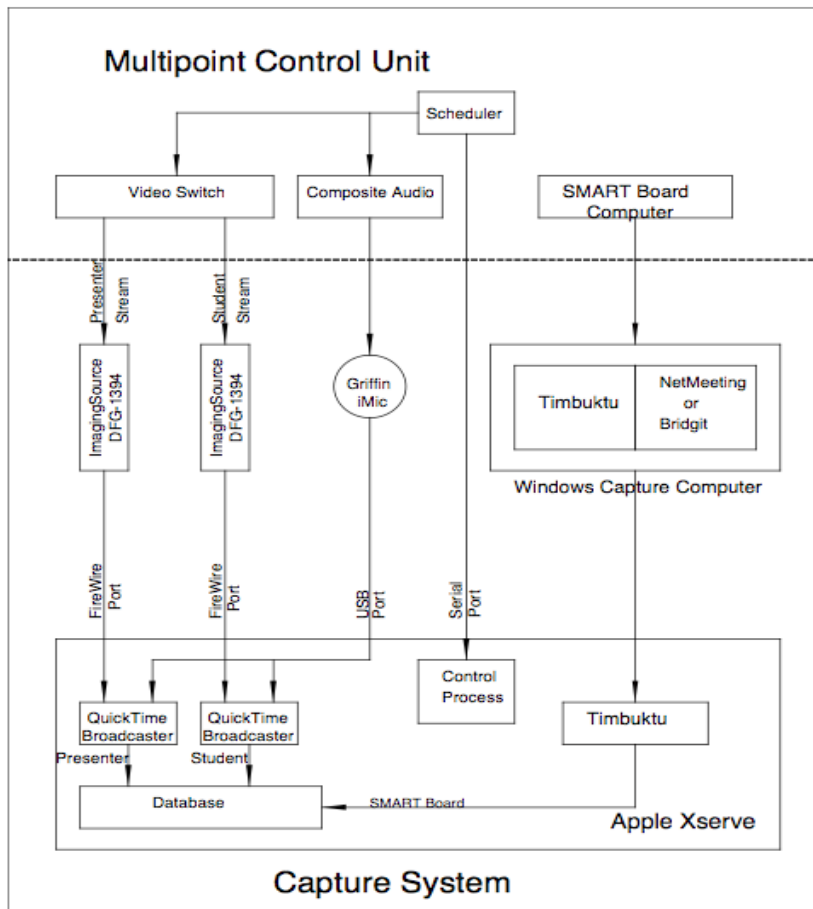


Figure 1.

Two DFG/1394-1 devices are used. One for the Presenter stream and one for the composite student stream. Each is connected to the XServe via a Firewire cable.

The audio encoding hardware selected is a Griffen Technology iMic USB Audio converter. This device captures the single audio stream provided by the MCU.

There is no special interface equipment required to collect the image from the local computer playing the smart board image.

The area at the bottom Figure 1 labeled Database is common to both the *Capture System* and the *Delivery System*. The collected material is stored in a directory and file structure on the XServe. To reference part of the data in this area a **<courseLayout>** convention is used.

The <courseLayout> convention is constructed as follows:

<courseIdentifier>/<streamIdentifier>/<termMonth>/<termDayTime>

where <courseIdentifier>

- is the name under which all material for a course is stored. It is arbitrary and is taken directly from commands provided by the "Scheduler",

<streamIdentifier>

- will be "lan" or "mod" representing 250 kbit per second and 50 kbit per second streams respectively,

<termMonth>

- will be formed as in September-2003. The sort order used for this field starts with September and ends with August of the next year,

<termDayTime>

- will be formed as DD-MM-YYYY-HHHH. i.e 20-08-2003-1352.

All files relating to a Capture session will be within the <termDayTime> directory.

The Control Process accepts requests from the scheduler, starting and stopping collection of audio, video and computer data as directed, and places the collected material into the files system according the <courseLayout> in preparation for the Delivery System to play in back on demand to the students.

1.4 Control Process Overview

The control process consists of a master process running continuously and several child processes that are started as needed to process collected material. An additional posting process is always running which responds to requests from the scheduler appearing on the RS232 control line from the MCU.

The Control process is driven by a single file representing the current state of the collection system and the status of collected material.

This is known as the Control file. The Control file contains status information for the various steps in a readable format to allow the operator to examine the status collection process at any time. A Web page is available from the XServe that will display the current Control file contents interpreting them as the current system state.

The Control process examines the Control file continuously looking for events that need to be processed. Those events are:

- Start a Capture process.
- Terminate a Capture process.
- Extend a Capture process
- Prepare the result of a Capture process for Web Delivery
- Prepare a modem variant for Web Delivery
- Perform end of day cleanup

Line 1 of the Control file contains the current status of a collection operation if one is being performed. This is the essence of the capture process and all means are used to ensure that it continues and is successful. The Control process will not terminate a Capture operation until the required collection time has expired.

Line 2 of the Control file is reserved for requests from the scheduler.

Line 3 of the Control file is used to control and maintain the status of the post processing operations required to make the collected material available for use by students.

Line 4 of the Control file is used to control and maintain the status of modem processing operations that provide the modem quality material for use by students.

Line 5 controls and maintains the status of cleanup and daily checking operations.

Line 6 and beyond in the Control file are used as a stack of post processing operations that need to be performed. These will be the events that lead to entries in lines 3, 4, and 5 of the Control file. The post processing of collected data does proceed in stages through the sequence of post, modem and cleanup.

The operations represented by lines 3, 4, and 5 can fail. The most likely cause is insufficient space to store collected material. In event that one of these operations fails the status of the event is added to the end of the Control file and it is marked as having failed and requiring operator intervention.

All operations that the Control process performs are logged into a file, LogFile, that may also be viewed from a Web Page. Each individual function logs distinct records along with a time stamp and an explanation of the event logged.

Post processing operations in general should not occur during the capture process. A configuration file, Config, is provided to allow specification of when these events start, the location of storage areas, and the level of information provided in the logging process.

1.5 Delivery Process Overview

The delivery of captured audio, video and computer data to students is controlled by a web page for high speed delivery and a web page for low speed delivery. These web pages are described in a separate document.



The scripts generating these web pages are aware of the <courseLayout> convention noted above. This convention is used to locate the streaming video reference pages which in turn invoke delivery of the streaming video from the Streaming Server.

In addition the these web pages provides a Bookmark capability for individual students allowing them to mark locations on the captured material that they may wish to refer to later. This process also uses the <courseLayout> convention.

1.6 The Details (Capture Process)

1.6.1 Location:

All files and scripts that are part of the control process are located in

/Library/QuickTimeStreaming/Control

on the XServe.

This location is embedded in each executable script in the variable \$CLOC.

1.6.2 Languages

Scripts are written in perl, /bin/shell and AppleScript. All of the Control scripts are written in perl but they invoke the others.

There are two compiled C++ programs used in the capture process

1.6.3 Files (in the order they are discussed below)

Control	- control file
LogFile	- logging file
Locks.pm	- inter process communication
Config	- changeable configuration items.
PostRS232	- The scheduler interface
PortSettings	- RS232 configuration
StrControl	- The master control script
StreamMonitor.pl	- Capture process
Start	- starts Broadcasters



S-S	- stops Broadcasters
ProcessMonitor.pl	- Post Processing
assembler	- video assembly program
preparer	- video preparation program
hintprefs.rtm	- hinting specifications
QTSettings	- compression specifications
ProcessModem.pl	- Modem Processing
ModemPrefs	- compression specifications
ProcessCleanup.pl	- end of day cleanup and checks
PostTest	- testing program

1.6.4 Control File

This file contains information pertaining to the sequence of events that are to be performed by the control system. It contains a minimum of 5 lines.

Line 1 represents the state of the Audio/Video/Smart collection process,

Line 2 is a repository for all requests received on the serial line from the scheduler

Line 3 represents the current state of post processing operations,

Line 4 represents the current state of modem processing operations,

Line 5 represents the current state of cleanup operations,

Line 6 and beyond contain queued work that has not been done.

This will contain

post processing operations, modem operations, cleanup operations, and

any operations that have failed.

Each line consists of multiple fields that change over time reflecting the status of the operation represented. Fields are separated by the string "<A>". Any blanks appearing within the fields are significant. When there is no capture in progress and no work pending the content of file should be as follows:

```
OP<A>IDLE
RQ<A>IDLE
PA<A>IDLE
PA<A>IDLE
```



The first field of each line is a check field maintained to ensure consistency of the file. The check field for line 1 is OP, line 2 is RQ and lines 3,4,5 take PA. Lines 6 and beyond have a different check field depending on the class of operation that is being stacked. These are PP for post processing, PM for modem processing, PC for cleanup process and FA for a stacked failure notice.

The second field of each line is a state field. During operation the state field, IDLE, portion of each of these lines is replaced by information relating to the operation being performed.

The details of these states, and the changes they go through, which may be of use when determining the cause of problems are explained in detail in the section StrControl below.

In each script that operates on the Control file there is logic to preserve previous copies of the file. The variable \$VL in each script (currently set to 6) governs the number of previous versions held. These files are hidden.

1.6.5 LogFile

All events that occur in the control process are logged in this file. There are 3 settable logging levels. The general format of an entry is:

```
[StrControl(V2.2)]<<2004/03/11 18:19.08>> [12018] --No Action taken
```

where [and] enclose the process script name and version,
<< and >> enclose the time the entry was made,
[and] enclose the process number of the task making
the entry and
-- precedes the actual log message

All inter process coordination is mediated by using locks on the LogFile. The rule is that the Control file may be changed only while the process maintains an exclusive write lock on the Logfile.



1.6.6 Locks.pm

This is a perl package implementing the lock management module.

1.6.7 Config

This file contains configuration information. During post processing operations information about the Streaming Server and Web Server configurations must be imbedded in the reference movies and streaming movies. That information is taken from this file.

The control scripts that use this file repeatedly read the contents to reset internal variables. In effect whenever you change a variable in the file the specified variable immediately changes in the scripts. Be careful!!

1.6.8 PostRS232 and PortSettings

PostRS232 is a simple perl script that waits for characters to be presented on the serial interface. PortSettings is a settings file that is used to configure the serial port.

When a line of data is received from the serial port the <ctrl 1>, <ctrl 2> characters are replaced with and <A> respectively and the command unchanged other than these field separators is placed in line 2 of the Control file.

<ctrl 3> from the serial port is a line terminator.

There are three possible commands that may appear on the serial line from the MCU. These are (after field separator changes): (Note the Start command normally is a single line of data.)

```
Start<A>cname<A>comment<A>tname<A>  
31<A>31<A>25<A>26<A>28<A>0<A>0<A>0<A>0<A>  
2004/01/29<A>10:50<A>5
```

```
Duration<A>90
```

```
Stop<A>
```



The first three fields in the Start command,

cname - name of the course,
comment - a comment field, and
tname - teacher name

are text fields entered in to the scheduler. Only cname is used and it becomes the <courseIdentifier> portion of <courseLayout>. The next nine fields consist of two digit codes specifying sites in the Ft.Vermillion system. The first of these is the originating site, the following eight are sites that are taking part in the current configuration. These fields are unused as well.

The last three fields identify the day, time and expected duration of the session.

The one argument to the Duration command replaces the duration in the ongoing operation.

1.6.9 StrControl

StrControl is a continuously running perl script. It examines the Control file every 12 seconds looking for work that it may be able to perform. It gives priority to the capture activity specified by line 1 of the Control file. The following summarizes the behavior of StrControl.

Lines 1 and 2 of the Control file are examined first to determine if there is ongoing capture activity.

Line 1 will progress through the following stages.

IDLE - no collection process,
STARTING - collection process is being organized,
ACTIVE - collection is happening
STAGE - a short cleanup following collection is occurring
IDLE - no collection process.

Line 2 may be in the stages IDLE, Start, Stop and Duration.

Start, Stop and Duration are entered into line 2 by the PostRS232 script.

StrControl responds to their presence and sets the state back to IDLE. When PostRS232 makes changes no checking is done, the new state is simply stored in the file. Note that this can only occur when the Control file is free for modification.

The following tests are made every 12 seconds.
When line 1 is in state IDLE and

a Start state (command) is in line 2, StrControl will build a request for a collection and place it in line 1. Line 1 is changed to state STARTING and StrControl starts the script StreamMonitor.pl and goes back to sleep for another 12 seconds.

Otherwise no action is taken, other than a log file entry is generated roughly every 25 minutes to the effect that "No Action taken".

When line 1 is in state STARTING and the StrControl awakes to look at the Control file it generates the message

No action taken. Collection process in transition

and goes back to sleep.

When line 1 is in state ACTIVE or STAGE and line 2 has state other than IDLE, when

a Start state (command) in line 2 StrControl writes the message

Collection is in Progress, Start request pending but is ignored.

to the log file and goes back to sleep. In this one case StrControl does not set line 2 to IDLE.

Or, when a Duration state (command) is in line 2, the new duration field is taken from the third field of the command and is inserted into the duration state field in line 1. (See the description of StreamMonitor.pl for details). This will cause capture session to be terminated. It has no effect if the state in line 1 is state STAGE.

Or when a Stop state (command) is in line 2, StrControl adds the message

Initiating Shutdown

to the log file and modifies the duration in line 1 to terminate the capture process. Again there is no effect when line 1 is in state STAGE.

Otherwise, i.e. Both line 1 and line 2 are in state IDLE, then StrControl will look beyond lines 1 and 2 and attempt to start the post processing operations.

The post processing operations all use one or more AppleScript driven Quicktime processes which must be serialized. For this reason none of the following will be done unless all of lines 1, 2, 3, 4 and 5 are in state IDLE.

Any one, and only one, stacked request in lines above 5 in the Control file that are eligible to run will be moved to the appropriate line 3, 4 or 5 in the Control file and the process for it will be started. This changes the state of the modified line to other than IDLE effectively blocking further post processing operations from being started until this process completes and the state of the line it uses is set back to IDLE.

Eligibility is determined by the order they are examined and the time value of the start time control fields PROC_START, MODEM_START and CLEANUP_START which are read from the Config file.

Note that the state that the stacked lines go through, (Field 1 contains PP, PM or PC), occur in sequence as the processes are completed. This ensures that all PP operations are completed before PM operations, which in turn are completed before PC operations.

StrControl will start capture operations running when there is an ongoing post processing operation. Although this is not a serious situation it is best to be avoided as the effect of the processing requirements of the post processing task may impact the quality of the captured video. The start time controls are provided to prevent this from happening.

PP operations are performed by ProcessMonitor.pl, PM operations are performed by ProcessModem.pl and PC operations are performed by ProcessCleanup.pl.



1.6.10 StreamMonitor.pl (and Start and S-S)

StreamMonitor.pl is a perl script invoked by StrControl to manage a capture session. It is started as a child and runs to completion for each capture session. It will in turn start child processes running the Apple QuickTime Broadcaster application to capture the combined audio and video streams from the MCU and AppleScript processes running the Timbuktu application to capture screen data from the Windows Capture Station

Before StreamMonitor.pl is invoked the StrControl script will set up parameters describing the capture request in line 1 of the Control file. Fields extracted from the start command in Line 2 are used to construct the capture request built in line 1 which appears as follows. (Note that in the control file it is a single line.)

```
OP<A>STARTING<A>YYYY/MM/DD  
HHMM<A>DD<A>AA<A>PP<A>  
cname<:>comment<:>2004/01/29<:>10:50<A>
```

where

OP - is a check field,

STARTING - is the state that the capture process is in at this point in time,

YYYY/MM/DD HHMM – represents the current time,

DD - is the required duration for the capture session in minutes,

AA – initially 0, is the actual time that capture has been operating,

PP – is unused, and

cname<:>comment<:>2004/01/29<:>10:50 – are fields extracted from the Start command in line 2.

When StreamMonitor.pl starts executing it does the following.

The content of line 1 of the Control file is checked and parameters required for the capture session are extracted from it.

In the situation that the directory represented by <courseIdentifier> in <courseLayout> does not exist it is created and a soft link to the same location is created in the movie capture volume. A notice that this has occurred is placed in the log file.

Any QuickTime Broadcaster applications running at this time are requested to stop by using the shell script s-s. If they fail to stop they are shutdown using the kill command.

Two copies of QuickTime Broadcaster are launched and configured to record compressed video from the two DFG/1394 encoders. Each also captures the audio stream provided by the iMic USB encoder. They are started from the shell script Start. They are given 90 seconds to complete startup and failing that the shutdown, kill and startup process will be repeated.

The Start script restores the preferences files of the QuickTime Broadcast01 and QuickTime Broadcas02 applications from back up files and then starts each application up as a GUI process. It then uses broadcast01ctl and broadcast02ctl to configure and start the capture process.

Recording is initialized and the recorded streams are immediately available for viewing via the DSS server and are simultaneously placed on disk for later viewing.

[Currently Disabled] Upon completion of the startup of Audio/Video capture the Smart Board capture process is started. This consists of an Apple Script running Timbuktu Pro in observation mode on the XServe , observing activity occurring on the screen of the Timbuktu Pro running on the Windows Capture Server, and saving that activity to a QuickTime Movie. The Apple Script used is built on demand and stored in the /tmp directory and run under osascript.

StreamMonitor.pl reports to the Control process that the capture process has started by changing the state field of line 1 from STARTING to ACTIVE.

StreamMonitor.pl examines the status of the recording process at 30 second intervals updating the AA field in line 1 until the preset duration time has elapsed. The capture process is considered complete when the AA field is larger than the DD field in line 1.

StreamMonitor.pl also monitors the size of the output files that the QuickTime Broadcaster applications are generating. It expects them to continue growing and to be of roughly similar sizes. When this is not the case it goes in to a restart mode using S-S and Start. It will continue to monitor and repeatedly shutdown and restart the capture process until the duration time has elapsed.

It does not monitor the growth of the file generated for the capture of the smart board activity.

When the accumulated recording time exceeds the duration time, i.e. AA exceeds DD in line 1 of the Control file, StreamMonitor.pl terminates the capture process.

Terminating the capture process involves shutting down the Timbuktu session and the two QuickTime Broadcaster applications. The state in line1 is changed to STAGE from ACTIVE and the collected data is moved to a staging area in <courseLayout>. A summary message noting the directories created is written to the log file.

When the staging operation is complete line 1 of the Control file is used to construct a post processing request which is added to the end of the Control file.

The format of this line is

PP<A>START<A>cname<A>Month-YYYY<A>DD-MM-YYYY-HHMM<A>

Where

cname is <courseIdentifier> ,

Month-YYYY is <courseMonth> , and

DD-MM-YYYY-HHMM is <termDayTime> of <courseLayout> .

The state of line 1 is set to IDLE and the StreamMonitor.pl script exits.



1.6.11 ProcessMonitor.pl (and assembler, preparer, hintprefs.rtm and QTSettings)

This script is invoked by StrControl to process a PP operation. When StrControl as described above determines that it is time run a post processing operation it places the appropriate request in line 3 of the Control file, changes PP to PA, and START to STARTING and invokes this script as a child process.

The command in line 3 is parsed to obtain information for the operation to be performed. At this time the directories for reference files will be created if they do not exist. This event will be noted in the log file.

Any running QuickTime Player applications running will be stopped. Effectively serializing QuickTime.

In line 3 STARTING will be changed to ACTIVE and the post processing operations will start.

Captured student movies are processed first, then instructor moves and then smart board movies.

The staging area

```
<courseLayout>/Stage
```

is examined for files that start with the text "student". If more than one such file is found then the post processing operation assumes that the capture session had been interrupted for some reason and the multiple movies will need to be combined into a single movie.

To do this it builds command to invoke the program assembler which

collects all the move segments starting with student,

verifies that they are valid movies,

sorts them into order by start times,

combines them in to a single movie with the missing time periods filled in with a filler movie

and saves the resultant new movie to <courseLayout> as a hinted movie.

assembler uses the file HintPrefs.rtm to obtain streaming specifications although these are the default QuickTime options for creating streaming files.

When only one movie is found in /Stage this step is skipped and that movie is copied to <courseLayout> without any changes.

A time stamp movie that is to be played on top of the student movie is constructed and stored in <courseLayout>. This movie is constructed using the program preparer. preparer uses the file HintPrefs.rtm.

A reference movie is built and placed in <courseLayout> in the web server area of the database. This is done by constructing an Apple Script program utilizing the QuickTime Player application which is stored in /tmp and run using osascript.

The above steps are then repeated for the "instructor" movies in <courseLayout>/Stage.

For the smart board movies an Apple Script is constructed to use QuickTime Player application to convert and compress the movie collected by Timbaktu Pro.

The original movie was built using the animation codec at 30 frames per second at the same size as the original windows screen. After this conversion process it will have the characteristics that are preset in the file QTSettings.

These setting are:

- Video
MPEG-4 Video, 320 wide, 240 high, millions of colours
12 frames per second, with a planned data rate of 249K bytes per second.
- Audio
MPEG-4 Audio, 11025 Hz, Mono
With a planned data rate of 1.9K bytes per second.

This process requires a significant amount of processing time.

preparer is then used to generate the timestamp movie for the smart board presentation.

A reference movie is then created using AppleScript and QuickTime Player application as for the student and instructor movies. It is also stored in the web server <courseLayout> portion of the database.

The content of line 3 is now placed at the end of the Control file with the first field set to PM and the second field set to START in preparation for the next processing step and line 3 is set to IDLE.

The script then exits.

1.6.12 ProcessModem.pl (and ModemPrefs)

This script is invoked by StrControl to process a PM operation. When StrControl as described above determines that it is time to run a post processing operation it places the appropriate request in line 4 of the Control file, changes PM to PA, and START to STARTING and invokes this script as a child process.

The command in line 4 is parsed to obtain information for the operation to be performed and STARTING in line 4 is changed to ACTIVE. At this time the directories for reference files will be created if they do not exist. This event will be noted in the log file.

AppleScript controlling the QuickTimePlayer application is used to build the modem file. The source is the already compressed MPEG-4 file built for the 250kbps stream which is recompressed according to the parameters set in the file ModemPrefs.

These settings are:

- Video
MPEG-4 Video, 160 wide, 120 high, millions of colours
4 frames per second, with a planned data rate of 39K bytes per second.
- Audio
MPEG-4 Audio, 8000 Hz, Mono
With a planned data rate of 1 K bytes per second.

A reference movie is constructed in the same manner in the web server <courseLayout> portion of the database.

The timestamp build in the post processing stage for the smart board capture is not currently laid on top of the smart movie.

Upon completion of the conversions to create the modem stream the content of line 4 is added to the end of the Control file with PA replaced by PC and ACTIVE replaced by START. The state of line 4 is set to IDLE and the script exits.

1.6.13 ProcessCleanup.pl

This script is invoked by StrControl to process a PC operation. When StrControl as described above determines that it is time run a post processing operation it places the appropriate request in line 5 of the Control file, changes PC to PA, and START to STARTING and invokes this script as a child process.

The command in line 5 is parsed to obtain information for the operation to be performed and STARTING in line 5 is changed to ACTIVE.

ProcessCleanup.pl runs through the directories setup by the previous scripts and checks that all the expected output files are in place. Should any of the checks performed by ProcessCleanup.pl fail the command is added back to the end of the Control file and is marked FA to indicate failure and the program exists. At this point only manual intervention can clean the error out of the Control file.

When these checks are completed successfully ProcessCleanup.pl removes files from <courseLayout>/Stage.

It then sets the state field for line 5 to IDLE and exits.

1.7 Streaming Server

The Streaming server configuration used is almost the default configuration for the Apple Darwin Streaming Server (DSS) Version 5.

The Streaming server is a Windows 2000 server with an SMB mount on a hidden mount point to the XServe where the Audio/Video and Smart board movies are stored.

Video is streamed on port 554

Files containing asynchronous sessions are located a directory specified by

it-vidcap.fvvd.ab.ca\MovieStorage\$\<courseLayout>

which maps directly on to the Database in Figure 1.

The student does not use a direct RTSP reference to the Streaming server, but instead launches the delivery from a reference page located at another hidden mount point also hosted on the XServe.

1.8 Web Server

The web interface is hosted on a Windows 2000 server using IIS extended to run perl scripts as CGI programs. There are two web pages provided on this server one for the 250kbps stream intended for use within the Ft. Vermillion system and one for the 50kbps stream intended for low speed access.

The reference movies required to link the web page to the streaming video are hosted on the Xserver at

it-vidcap.fvvd.ab.ca\WebStorage\$\<courseLayout>

Authentication of students and access to the web interface is controlled by the IIS Server.

Some space on this server is required to provide a database of Bookmarks to the movies for student use. The structure of this database follows the <courseLayout> convention.

A full description of the web pages is provide in "Automatic Recording and Streaming of Synchronous Distance Education Classes"
Montgomery et al.

1.9 Administration Tools

Two summary web pages are available from a web server running on it-vidcap. These are intended for operator use.

Event	Action	Start Time	Duration	Actual Time	Original Command Arguments
Collection	IDLE				
Request	IDLE				
			Course	Month	Period
Post Processing	IDLE				
Compression	IDLE				
Cleanup	ACTIVE		XStuff	March-2004	15-03-2004-1244

The first of these pages at <http://it-vidcap.fvvd.ab.ca/cgi-bin-Summary.cgi> shows a snapshot of the control file. It is set to refresh on 5 second intervals. The second one <http://it-vidcap.fvvd.ab.ca/cgi-bin/WatchLog.cgi> shows the last few lines of the log file. It refreshes on 30 second intervals.

The Last 20 log file entries IN REVERSE ORDER
[StrControl(V2.2)]<<2004/03/15 16:02.18>> [12018] --No Action taken
[StrControl(V2.2)]<<2004/03/15 15:35.34>> [12018] --No Action taken
[StrControl(V2.2)]<<2004/03/15 15:08.51>> [12018] --No Action taken
[StrControl(V2.2)]<<2004/03/15 14:42.07>> [12018] --No Action taken
[StrControl(V2.2)]<<2004/03/15 14:15.24>> [12018] --No Action taken
[StrControl(V2.2)]<<2004/03/15 13:48.41>> [12018] --No Action taken
[StrControl(V2.2)]<<2004/03/15 13:21.58>> [12018] --No Action taken
[StrControl(V2.2)]<<2004/03/15 12:55.14>> [12018] --No Action taken
[StrControl(V2.2)]<<2004/03/15 12:48.45>> [8760] --ProcessCleanup.pl terminated
[CleanupProcess(V2.2)]<<2004/03/15 12:48.44>> [8763] --Process Terminating with prejudice.
[CleanupProcess(V2.2)]<<2004/03/15 12:48.44>> [8763] --FAILED The required movies have not been generated.
[CleanupProcess(V2.2)]<<2004/03/15 12:48.44>> [8763] --Final Cleanup and check for XStuff, March-2004, 15-03-2004-1244 starting.
[StrControl(V2.2)]<<2004/03/15 12:48.44>> [12018] --Started ProcessCleanup.pl
[StrControl(V2.2)]<<2004/03/15 12:48.34>> [8692] --ProcessModem.pl terminated
[ModemProcess(V2.2)]<<2004/03/15 12:48.34>> [8695] --Modem reduction of XStuff, March-2004, 15-03-2004-1244 files completed.
[ModemProcess(V2.2)]<<2004/03/15 12:48.34>> [8695] --Script <> running as [8724] terminated.
[ModemProcess(V2.2)]<<2004/03/15 12:48.33>> [8695] --Script <> running as [8724]
[ModemProcess(V2.2)]<<2004/03/15 12:48.33>> [8695] --Creating Reference Movie
[ModemProcess(V2.2)]<<2004/03/15 12:48.33>> [8695] --Script <> running as [8720] terminated.
[ModemProcess(V2.2)]<<2004/03/15 12:48.33>> [8695] --Script <> running as [8720]

1.9.1 Outstanding issues or problems.

The Apple Broadcaster application while intended to operate in background mode has been unreliable in that mode in all tests. For this reason it is currently being operated as foreground application. I consider this unacceptable, and have been trying to get Apple to correct this situation.

A known problem with QuickTime Broadcaster 's estimation of space exists when storing captured video requires that the Broadcaster application be set up with it's own small drive (less that 4 Gigabytes) in size. This does complicate some of the work that the Control process does.

The Script StrControl is unable to function as a detached process. I suspect that this relates to the launch process used by Apple for applications such as the QuickTime Broadcaster. I would expect that resolution of issue 1 would resolve this one as well.

The specifics of what the system and or operator should do in response to a state of FA have yet to be defined. At this point recovery from error will require that the operator understand the body of the perl scripts.

There is no mechanism for setting up or managing the <courseLayout> structures. Current practice is to do this from the text strings entered to the scheduler.

There is no mechanism for starting and stopping the Audio/Video/Smart capture process from other than the scheduler. Note that this was a design decision.

At the time this document was written the mechanism to capture the smart board activity is INACTIVE and generally untested.

1.10 Summary

With respect to the original design goals.

- The preferred equipment was to be oriented around the Windows operating system



Partial success. The choice of Macintosh equipment for the capture process was driven primarily by the choice of QuickTime MPEG-4 for the delivery mechanism.

The only viable application for providing this process from a Windows based computer was the Sorenson Broadcaster. Sorenson withdrew their product from the market during this project.

- The system was to operate without operator intervention for daily operation.

This goal was attained.

- Existing facilities and software were to be used. Special purpose solutions were to be minimized.

Two special applications were written to accomplish specific video manipulations. These were described in this document. assembler is used to patch together chunks of video into a continuous stream preserving the elapsed time of the capture process. This was accomplished by inserting filler pieces into the movie. preparer was built to construct a time stamp overlay movie for a movies of any length based on the start time from the movie.

The perl language and Apple Script were used to glue all the applications together into a composite whole. In retrospect it might have been better to build a single application that performed the complete capture process as a single step although a development of this scope might well have been beyond the resources available.

- Internal to the Ft Vermillion school district bandwidth requirements were to be kept to 250 kilobits per second.

This goal was attained.

- A modem level stream was to be produced.

This goal was attained.



- The capture system was to be controlled exclusively by the "Scheduler" which also controls the VLPE.

This goal was attained.

- Access to the collected material was to be restricted to students in the Ft. Vermillion system.

This goal was attained.

- The captured material was to be available to students during the process of capturing it.

Partially attained. The presenter stream and student stream are available in this manner. The video stream for the smart board computers is not available at this time. However there are other ways of sharing the image on the smart board.

This requirement was in fact one of the most restricting ones in that it dictated the use of the Apple QuickTime Broadcaster applications which is the only available of the shelf tool capable of performing this function.