Appendix E

Videoconferencing Application

The following section will discuss two different types of video conferencing applications used in the LearnCanada Project. It will discuss the ISABEL application and the H.323 protocol applications/systems.

(i) ISABEL Application

The following section will describe the ISABEL application, development changes from the start of the project toward the end of the project, and some of is advantages over the H.323 system.

Description:

The ISABEL application is a collaborative video conferencing tool developed in Spain at the Universidad Politécnica de Madrid. The application is a Computer Supported Collaborative Workspace (CSCW). The more recent versions of the application run on the Linux OS. Development and testing is mainly done on SUSE, however, for the LearnCanada Project, we opted to use Redhat instead.

In order to run an ISABEL session, one site must launch the application as the master site. (Instructions on how to start ISABEL as the master, see appendix G.) Once the master is launched, all other participating site would join the master.

There are three modes of operations, tele-conference, tele-meeting, and tele-class. The tele-conference mode is designed for a large number of sites where each site may have a number of viewers, ie; an auditorium. All controls such as video, audio, or tools are given to the master site during startup. Minimal controls can be given to participating sites, but are done by the master. This makes for a more structured, less chaotic event.

The tele-meeting mode is for a smaller group, ie; small meeting involving 2 to 4 people. In this mode of operation, controls are open to all participating site. A completely open environment compared to the tele-conference. This can pose a problem is there are too many people trying to take control.

The tele-class mode fits in between the other two modes. Some controls are open to all sites, however, there is still a sense of structure. The site who currently has control, can give over control when they are finished to another site. There is still however, a master controller.

Another feature with ISABEL is the flowserver, similar to a Multi Control Unit (MCU). The ISABEL flowserver has several functions. One is to bridge networks together, ie; can be used to bridge a unicast environment with a multicast environment. A second function would to lighten the CPU load of the Interactive ISABEL workstation during a session or event.

ISABEL Versions and Features:

From the beginning of the project, the ISABEL version used was 4.3. As the project progressed, new builds and new beta versions became available. The final version used by the LearnCanada project was 4.5Beta12. Improvements with the audio codecs, video codecs, and collaborative tools were significant. One feature which was added to the latter versions, was the sEssion Development Language (EDL). The EDL is the Web Portal which will be described in the next section "ISABEL Web Portal". Shared Application (however, not 100% stable) was also a new feature which make use of Virtual Network Computing

(VNC) servers. Graphical User Interface (GUI) for setup up and configuring ISABEL was changed to make it more user friendly.

As Linux becomes more advanced, drivers are becoming more and more readily available. The requirements for ISABEL is very simple, if the hardware is supported by Linux then there is a very good chance that the hardware is supported by ISABEL. There is however a few hardware components that you to keep in mind and those are the audio card and the video capture/tuner card.

The video capture card that are used must be supported by video4linux. Most recommended cards for the LearnCanada project are cards which use the Brooktree BT848 or BT878 chipset. An alternative camera source which is also supported by ISABEL is USB camera. Again, as long as the camera is supported by video4linux, it will run on ISABEL.

The application also requires the use of Advanced Linux Sound Architecture (ALSA) drivers for the sound card. OSS drives have been known to work, however, the ISABEL audio component is optimized for ALSA. The reason for using the ALSA vs OSS drivers is the flexibility of ALSA. ALSA has the capability of emulating full duplex when the hardware does not support full duplex.

Technical Web Site

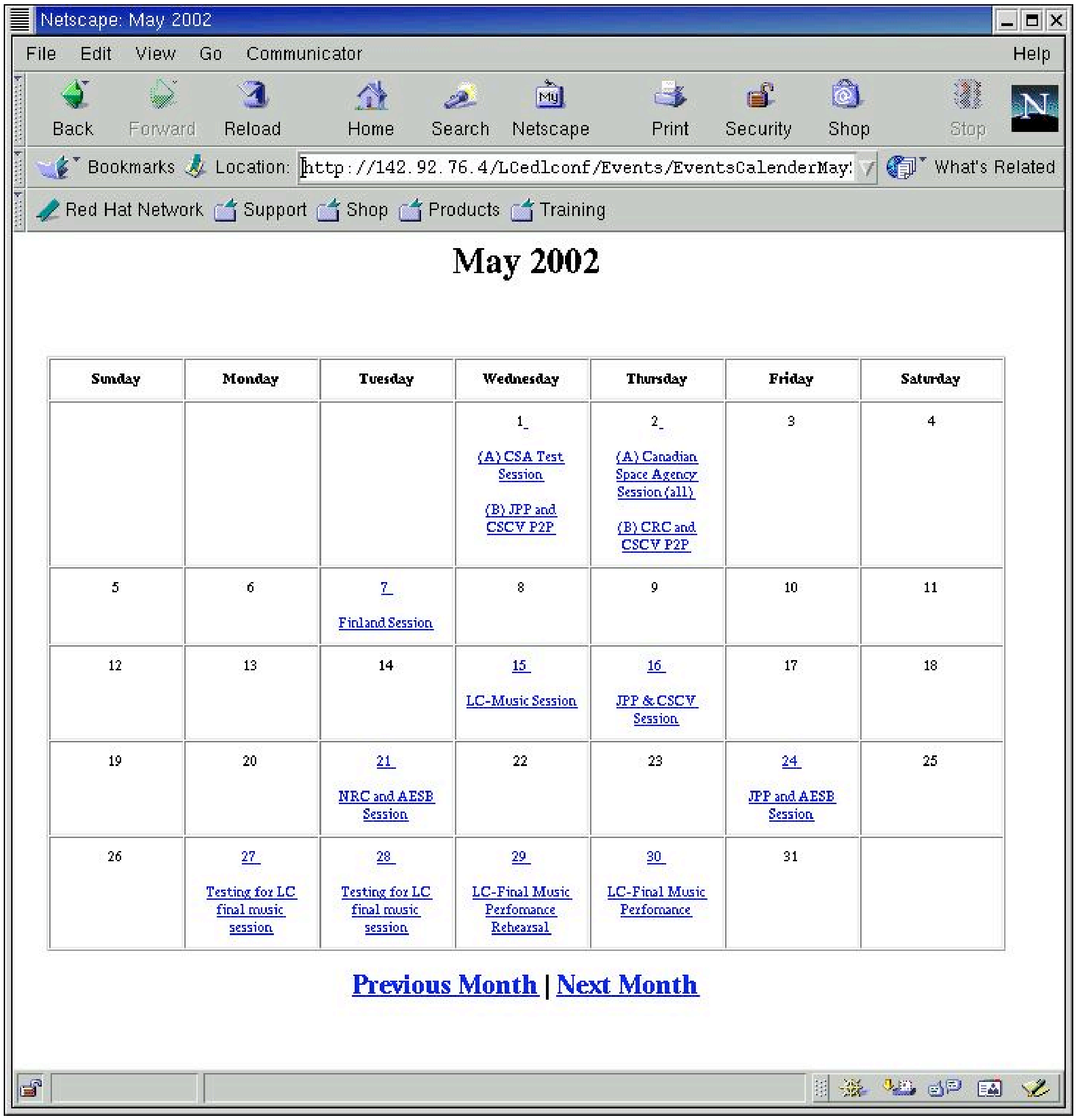
The technical web site was setup on http://142.92.71.29 as a repository for technical documentation, user guides/manuals, software, software updates, calendar of events, web portal, etc. The next few sections will describe the ISABEL Web Portal created on the technical web site.

ISABEL Web Portal

The later versions of the ISABEL application featured the Web Portal. The Web portal allowed for quick and easy ISABEL startups.

A portal for ISABEL has been setup on the BADLABs technical Web Site for the LearnCanada project. The Portal is setup on http://142.92.71.29, (or http://142.92.76.4). This machine is setup as a Web Server, running apache. The portal allows end users to access the above URL, to view a list of scheduled meetings, test schedules, and scheduled events, and when they will be taking place. The user can then join the session if they wish by clicking on the appropriate link. An .isa file will then be downloaded from the server to the local machine and automatically launch the ISABEL application. The user will then be prompt to key in their login name. Simply hit the start icon to join the session. The login names are provided to all sites before hand. Each login name is associated with it's own IP address. It is not possible to go onto any machine configured with Linux and ISABEL and simply enter the individual's login name.

The diagram below illustrates the event calendar with various scheduled events.


Netscape: May 2002
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://142.92.76.4/LCedlconf/Events/EventsCalenderMay!
What's Related
Red Hat Network Support Shop Products Training

May 2002

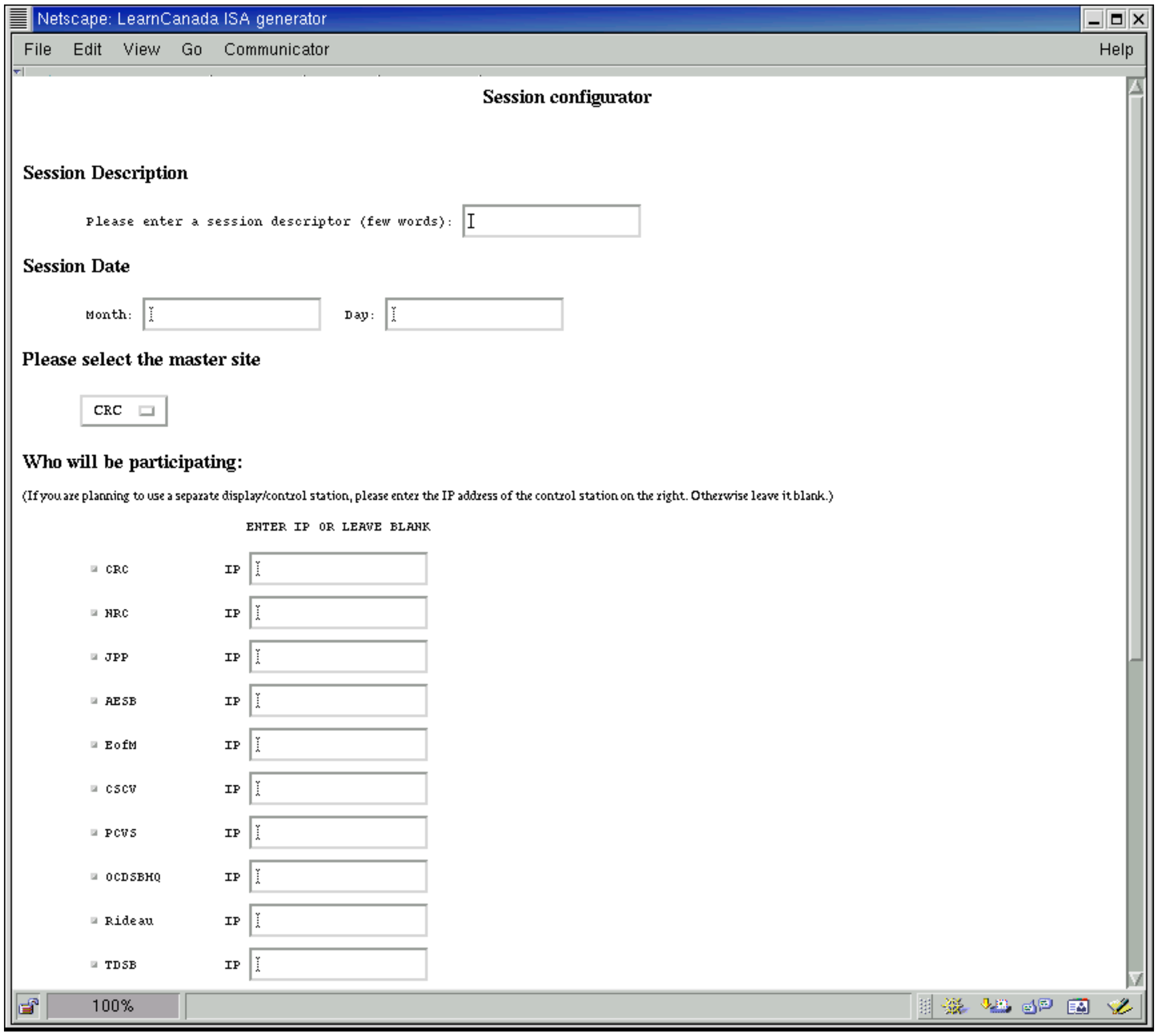
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
			1 (A) CSA Test Session (B) JPP and CSCV P2P	2 (A) Canadian Space Agency Session (all) (B) CRC and CSCV P2P	3	4
5	6	7 Finland Session	8	9	10	11
12	13	14	15 LC-Music Session	16 JPP & CSCV Session	17	18
19	20	21 NRC and AESB Session	22	23	24 JPP and AESB Session	25
26	27 Testing for LC final music session	28 Testing for LC final music session	29 LC-Final Music Perfomance Rehearsal	30 LC-Final Music Performance	31	

Previous Month | Next Month

User generated .isa files:

The purpose of the web portal was to make life easier for teachers/users to start an ISABEL session without the assistance of a technical staff. It was found that this the web portal indeed made life easier for the end user, however, this was not the case for some. As the LearnCanada project progressed, concurrent point to point sessions became more frequent. Meetings over ISABEL would be scheduled a day in advanced. In order to add the meeting to the calendar of events, CRC was notified, and the ISA files would be created, and the calendar would be updated.

Because of the number of concurrent sessions were growing, it became apparent that there was a need to allow the teachers/users to create sessions, ie; generate the ISA files, themselves. On the BADLAB's LearnCanada Technical Website, a session generator was created. Written in Perl to run under CGI, was a program to allow the end users to type in essential data needed to generate the ISA file. A form would be filled out asking the user for specific information needed for the program to generate the ISA file.


Netscape: LearnCanada ISA generator
File Edit View Go Communicator Help
100%

Session configurator

Session Description

Please enter a session descriptor (few words):

Session Date

Month: Day:

Please select the master site

CRC

Who will be participating:

(If you are planning to use a separate display/control station, please enter the IP address of the control station on the right. Otherwise leave it blank.)

ENTER IP OR LEAVE BLANK

CRC	IP
NRC	IP
JPP	IP
AESB	IP
EofM	IP
CSCV	IP
PCVS	IP
OCDSBHQ	IP
Rideau	IP
TDSB	IP

Once the information is inputted, the user submits the query and the programs writes the data to a temporary file called isafile.out. A verification screen appears and prompts the user if all data is correct. The following screen will ask the user to hit enter and create the ISA file and store it on the server.

The following is coding for the ISA file generator.

LC-Form.html -> lcisa.cgi -> lcisa2.cgi -> ISAGenerator.cgi

Code for LC-Form.html

This is the form where all the necessary data must be entered.

```
<HTML><HEAD><TITLE>LearnCanada ISA generator</TITLE></HEAD>
```

```
<BODY>
<form action="http://142.92.76.4/cgi-bin/lcisa.cgi" method="POST">
<center><h1>Session configurator</h1></center><br><br>
<H1>Session Description</H1>
<PRE>
        Please enter a session descriptor (few words): <input type="text" name="descriptor">
</PRE>

<H1>Session Date</H1>
<PRE>
        Month: <input type="text" name="Month">   Day: <input type="text" name="Day">
</PRE>

<h1>Please select the master site</h1>
<ul>
        <select name="Mastersite">
        <option value="CRC"> CRC
        <option value="NRC"> NRC
        <option value="JPP"> JPP
        <option value="AESB"> AESB
        <option value="EofM"> EofM
        <option value="CSCV"> CSCV
        <option value="PCVS"> PCVS
        <option value="OCDSBHQ"> OCDSB
        <option value="Rideau"> Rideau
        <option value="TDSB"> TDSB
        </select>
</ul>

<H1>Who will be participating:</H1>
(If you are planning to use a separate display/control station, please enter the IP address of
the control station on the right.  Otherwise leave it blank.)
<BR>
<PRE>
                            ENTER IP OR LEAVE BLANK
</PRE>
<PRE>
        <input type="checkbox" name="CRC" value=1> CRC          IP <input type="text"
name="displayCRC">
        <input type="checkbox" name="NRC" value=1> NRC          IP <input type="text"
name="displayNRC">
        <input type="checkbox" name="JPP" value=1> JPP          IP <input type="text"
name="displayJPP">
        <input type="checkbox" name="AESB" value=1> AESB                IP <input type="text"
name="displayAESB">
        <input type="checkbox" name="EofM" value=1> EofM                IP <input type="text"
name="displayEofM">
        <input type="checkbox" name="CSCV" value=1> CSCV                IP <input type="text"
name="displayCSCV">
        <input type="checkbox" name="PCVS" value=1> PCVS                IP <input type="text"
name="displayPCVS">
        <input type="checkbox" name="OCDSBHQ" value=1> OCDSBHQ  IP <input type="text"
name="displayOCDSBHQ">
        <input type="checkbox" name="Rideau" value=1> Rideau            IP <input type="text"
name="displayRideau">
        <input type="checkbox" name="TDSB" value=1> TDSB                IP <input type="text"
name="displayTDSB">
</PRE>
```

```
<H1>Separate Flowserver: </H1>(Recommended if there is more than 4 sites participating)
<UL>
        <select name="flowserver">
        <option value="NO"> NO
        <option value="YES"> YES
        </select>
</UL>

<H1>Service</H1>
<ul>
        <select name="Service">
        <option value="Telemeeting"> Tele-Meeting
        <option value="Teleconference"> Tele-Confernece
        </select>
</ul>

<H1>Traffic shaping</H1>
<PRE>
        Upload bandwidth (kbps):       <input type="text" name="upbw">
        Download Bandwidth (kbps):      <input type="text" name="downbw">
</PRE>

<br><br><center><input type="submit"></center>

</form>
</BODY></HTML>
```

Code for lcisa.cgi

This file takes the data from the LC-Form and generates a temp cale called isafile.out. It re-reads the file and verifies to the user that the correct data is entered. If all data is correct, the next screen will be lcisa2.cgi

```
#!/usr/bin/perl
print "Content-type:text/html\n\n";

read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
@pairs = split(/&/, $buffer);
foreach $pair (@pairs) {
    ($name, $value) = split(/=/, $pair);
    $value =~ tr/+/ /;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
        $value =~ s/\n/ /g;     # replace newlines with spaces
        $value =~ s/\r//g;      # remove hard returns
        $value =~ s/\cM//g;     # delete ^M's
    $FORM{$name} = $value;
}

open(OUTF,">isafile.out") or dienice("Couldn't open isafile.out for writing: $!");

flock(OUTF,2);
seek(OUTF,0,2);
print OUTF "$FORM{'descriptor'}|";
print OUTF "$FORM{'Month'}|";
print OUTF "$FORM{'Day'}|";
print OUTF "$FORM{'Mastersite'}|";
print OUTF "$FORM{'Service'}|";
print OUTF "$FORM{'flowserver'}|";

print <<EndHTML;
<html><head><title>Results</title></head>
<body>
<Center><h1>The following parameters will be used to create the session</h1></Center>
<br>
<b>Session Description: </b> $FORM{'descriptor'}
<br>
<b>Session Date: </b> $FORM{'Month'}  $FORM{'Day'}
<br>
<b>Master Site: </b> $FORM{'Mastersite'}
<br>
<b>Flowserver: </b> $FORM{'flowserver'}
<br><br>
EndHTML
;

%boxes = ("CRC" => "CRC",
          "NRC" => "NRC",
          "JPP" => "JPP",
          "AESB" => "AESB",
          "EofM" => "EofM",
          "CSCV" => "CSCV",
          "PCVS" => "PCVS",
          "OCDSBHQ" => "OCDSBHQ",
          "Rideau" => "Rideau",
          "TDSB" => "TDSB" );

$sitecounter = 0;
foreach $key (keys %boxes) {
     if ($FORM{$key} == 1) {
                  $sitecounter++;
          print OUTF "$key,";
     }
}

print "<b>Participating sites:</b><br><br>\n";
foreach $key (keys %boxes) {
   if ($FORM{$key} == 1) {
          print "$boxes{$key}<br>\n";
   }
}

print OUTF "|$FORM{'upbw'}|$FORM{'downbw'}";
close(OUTF);

print <<EndHTML;
<br>
<b>Traffic shaping parameters:</b><br><br>
Upload Bandwidth: $FORM{'upbw'} kbps<br>
Download Bandwidth: $FORM{'downbw'} kbps
<br><br>
<hr>
<br><Center>
If this is incorrect, please go back and re-enter the correct information.
```

```
<br><br>
If this is correct, please continue.
<br></Center>
<form action="http://142.92.76.4/cgi-bin/lcisa2.cgi" method="POST">

<br><center><input type="submit"></center>
</form>
EndHTML


print <<EndFoot;
</body></html>
EndFoot
;
```

Code for lcisa2.cgi

Final page before sending the isafile.out for generation of the .isa file.

```
#!/usr/bin/perl
print "Content-type:text/html\n\n";

print <<EndHTML;
<html><head><title>ISA file creation</title></head>
<body>
<h2 align="CENTER">Your ISA file is being generated....</h2>
<br>
<CENTER><A HREF="ISAgenerator.cgi">Continue</A></CENTER>
<br>
EndHTML

print <<EndFoot;
</body></html>
EndFoot
;
```

Code for ISAGenerator.cgi

This file will generate and store the ISA file on the server.

```
#!/usr/bin/perl
print "Content-type:text/html\n\n";

read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
@pairs = split(/&/, $buffer);
foreach $pair (@pairs) {
   ($name, $value) = split(/=/, $pair);
   $value =~ tr/+/ /;
   $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
      $value =~ s/\n/ /g;     # replace newlines with spaces
      $value =~ s/\r//g;      # remove hard returns
      $value =~ s/\cM//g;     # delete ^M's
   $FORM{$name} = $value;
}

#This section will read the session information generated by the user.
```

```
open(INF,"isafile.out") or dienice("Couldn't open isafile.out for reading: $! \n");
$data = <INF>;
close(INF);
open(INF,"address.out") or dienice("Couldn't open address.out for reading: $! \n");
@address = <INF>;
close(INF);

$sitecounter = 0;
%sites = ();
%site = ();
%site_name = ();
$site_ip = ();
$Mastersite = 0;
$master_ip = 0;
$master_code = 0;
$Service = 0;
$x = 0;

%boxes = ("CRC" => "CRC",
        "NRC" => "NRC",
        "JPP" => "JPP",
        "AESB" => "AESB",
        "EofM" => "EofM",
        "CSCV" => "CSCV",
        "PCVS" => "PCVS",
        "OCDSBHQ" => "OCDSBHQ",
        "Rideau" => "Rideau",
        "TDSB" => "TDSB" );


#This section will look at the isafile.out file.
foreach $i ($data) {
          chomp($i);
          ($descriptor,$Month,$Day,$master,$service,$flowserver,$boxes,$upbw,$downbw) =
split(/\|/,$i);
          @invlist = split(/,/,$boxes);
          foreach $j (@invlist) {
                    $site{$x} = $j;
                    $x++;
          }
}

#This session will look at the address.out file.
foreach $i (@address) {
      chomp($i);
      ($code,$public_name,$ip) = split(/\|/,$i);
          $z = 0;
          while ($z <= $x) {
                    if ("$site{$z}" eq "$code") {
                              $site_code{$z} = $public_name;
                              $site_ip{$z} = $ip;
                              $z = $x;
                    }
                    else {
                              $z++;
                    }
          }
          if ("$master" eq "$code") {
```

```
                        $master_ip = $ip;
                        $master_code = $public_name;
            }
}

#This section will create the edl format isa file.
open(OUTF,">session.isa") or dienice("Couldn't open isafile.out for writing: $!");

flock(OUTF,2);
seek(OUTF,0,2);
            print OUTF "#---------- ISA file generator ----------\n";
            print OUTF "#Session date: $Month/$Day\n";
            print OUTF "\n";
            print OUTF "BEGIN: EDL\n";
            print OUTF "VERSION: 1.5\n";
            print OUTF "BEGIN: SESSION\n";
            print OUTF "ID: LearnCanada\n";
            print OUTF "DELIVERY_PLATFORM: Isabel\n";
            print OUTF "SERVICE: $service\n";
            print OUTF "QUALITY: 2M\n";
            print OUTF "SESSION_PORT: 32000\n";
            print OUTF "MASTER: $master\n";
            print OUTF "OPEN_SESSION: yes\n";
            print OUTF "ALLOW_WATCHPOINTS: no\n";
            print OUTF "DESCRIPTION: $descriptor\n";
            print OUTF "SITES: $master";
$y = 0;
while ($y < $x) {
            print OUTF " $site{$y}";
            $y++;
}
            print OUTF "\n";
            print OUTF "END:SESSION\n";
            print OUTF "\n";

#Creates an EDL site for the master.
        print OUTF "#<---------- $master --------------->\n";
        print OUTF "BEGIN: SITE\n";
        print OUTF "ID: $master\n";
        print OUTF "PUBLIC_NAME: $master_code\n";
        print OUTF "SITE_ADDRESS: $master_ip\n";
        print OUTF "ROLE: interactive\n";
        print OUTF "CONNECTION_MODE: mcu\n";
        print OUTF "MCU_ADDRESS: \n";
        print OUTF "BANDWIDTH: 2000\n";
        print OUTF "NETWORK_ACCESS: Ethernet\n";
        print OUTF "END: SITE\n";
        print OUTF "\n";

#This will loop to create the individual sites
$y = 0;
while ($y < $x) {
            print OUTF "#<---------- $site{$y} --------------->\n";
            print OUTF "BEGIN: SITE\n";
            print OUTF "ID: $site{$y}\n";
            print OUTF "PUBLIC_NAME: $site_code{$y}\n";
            print OUTF "SITE_ADDRESS: $site_ip{$y}\n";
            print OUTF "ROLE: interactive\n";
```

```
		print OUTF "CONNECTION_MODE: mcu\n";
if ($flowserver eq "no") {
		print OUTF "MCU_ADDRESS: $master\n";
}
else
{
		print OUTF "MCU_ADDRESS: 142.92.76.4\n";
}
		print OUTF "BANDWIDTH: 2000\n";
		print OUTF "NETWORK_ACCESS: Ethernet\n";
		print OUTF "END: SITE\n";
		print OUTF "\n";
		$y++;
}

#This is the final line to close the ISA file
print OUTF "END: EDL\n";
close OUTF;

print <<EndHTML;
<html><head><title>Thank You</title></head>
<body>
<Center>
<h1>Thank You!</h1><br>
<h2>Your ISA file has been generated!</h2></Center>
<br><br>
<Center><a href="http://142.92.76.4/index.html">Return to Home</a><p></Center>
<br><br>
<Center><a href="http://142.92.76.4/cgi-bin/Calendar/EventsCalendarJune2002.cgi">Return
to Calendar</a><p></Center>
</body></html>
EndHTML

sub dienice {
	my($msg) = @_;
	print "<h2>Error</h2>\n";
	print $msg;
	exit;
}
```

<u>Technical Setup:</u>

How to configuring the isa configuration files.

The isa files is a text file with the following format.

Anything after a "#" is not read in. All variables must be contained within a "BEGIN" and "END" statement. For example, individual site variables must each be contained within it's own "BEGIN / END" statements. All sites variable and global variables must be contained within another "BEGIN / END" statement.

Site variables that must be included in the isa file are:

ID:	= Site code
PUBLIC_NAME:	= Description of the Site
SITE_ADDRESS:	= IP address of the ISABEL machine
ROLE:	= Interactive site, master, flowserver

CONNECTION_MODE:	= Point to point, multicast or flowserver
MCU_ADDRESS:	= IP address of the Flowserver
BANDWIDTH:	= Traffic shaping value
NETWORK_ACCESS:	= Ethernet, ISDN

Other variables and options are can also be entered, however, will not be included here.

Configuring the Netscape Browser to access the web portal:

For a site to join an ISABEL session via the web server, the Linux station must have the new version of ISABEL installed, ie; version 4.5-x. As well, the Netscape browser must be configured. To configure the browser, go to

Edit -> Preference -> Applications

Select NEW and add the following parameters:

Description:	Isabel Web Service
MIMEType:	application/risabel
Suffixes:	.isa

Note: make sure you enable the "Use this MIME as the outgoing default for these extensions."

Select the applications and enter the following:

```
/usr/local/isabel/bin/isabel -edlconf %s
```

Advantages and Disadvantages:

From the LearnCanada project, we have learned that there are many advantages of the ISABEL application over the H.323 systems. Some of the advantages are the collaborative tools allowing users to shared and edit documents, slide presentation feature, and capability to run higher quality video with multiple sites. As well, the new portal feature, which has been incorporated into the new versions of ISABEL, is an advantage. Another advantage would be the flowserver.

Some of the disadvantages comes from using the Linux OS. If the end user has little or no experience with Linux, they usually consider this to be a big disadvantage.

(ii) H.323

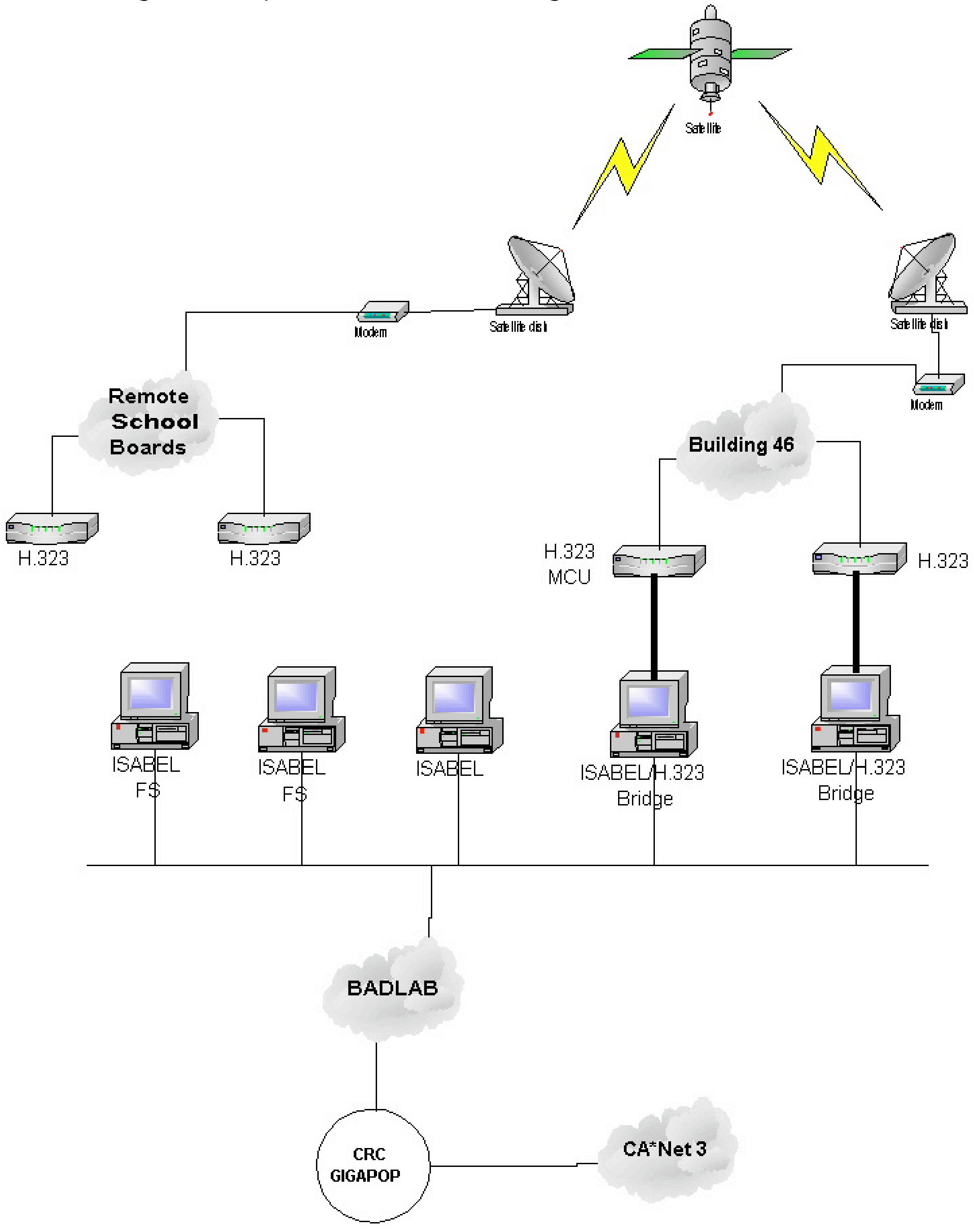
H.323 is an alternative solution for videoconferencing. H.323 uses IP protocol. For the LearnCanada project, the H.323 was used to reach areas where bandwidth was more limited, ie; remote sites connecting via satcom.

The H.323 units used in the project is capable of running H.261 or H.263 video protocol. Audio codecs range from G.711, G.722 to G.723.

The H.323 system used for the project was the Polycom FX. This system was equipped with a built in 4 point MCU, capable of 2 Mbps in a point to point session and 386 Kbps in a four point session. The FX was also used to stream H.263 video, ie; ISABEL session, over the network at 512 Kbps.

It was also possible to bridge H.323 and the ISABEL application together (with low quality video). For this, a video card with a composite or s-video output was needed. The output of the display would be fed into the input signal of the H.323 unit, in this case the Polycom station. The video and audio output of the Polycom would then feed into the input of the video capture card and audio line in.

Network diagram: Setup for H.323 to ISABEL bridge.



In order to bridge the H.323 with ISABEL, a dummy ISABEL workstation was needed for each H.323 system. The video output from the H.323 system would be feed into the input of the ISABEL video capture card. The ISABEL station was installed with a video buffer card equipped with a TV composite and s-video (640x480 resolution) output. The output of this card would be feed into the video input of the Polycom. The audio was setup in the similar manner with the audio output of the polycom was connected to the line input of the

ISABEL audio card. The line out of the audio card was connected to the audio input of the Polycom.