Assignment - 2
High Speed Networks (3CS1101)
Roll No: 16MCEC12
Name: PATEL AKSHAR B.

NS – 2:

Ns is a discrete event simulator targeted at networking research. Ns provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks.

EvalVid:

EvalVid is a framework and tool-set for evaluation of the quality of video transmitted over a real or simulated communication network. It is targeted for researchers who want to evaluate their network designs or setups in terms of user perceived video quality.

1. Integration of NS-2 and EvalVid:

Raw video can be encoded by the required codec using tools provided by EvalVid. Video traces that contain information about video frames, like the frame type, time stamp, etc, are generated by using EvalVid. These trace files are used as a traffic source in ns-2.

Scripting an agent in ns-2:

Agents created in ns-2 need to read the generated video trace file. This agent code is easily available on the Internet. The following steps must be taken for integrating the agent code in ns-2.

Step 1: Put a frametype_ and sendtime_ field in the hdr_cmn header. The frametype_ field is to indicate which frame type the packet belongs to. The I frame type is defined to 1, P is defined to 2, and B is defined to 3. The sendtime_ field records the time the packet was sent. It can be used to measure end-to-end delay.
You can modify the file packet.h in the common folder as follows:

```
struct hdr_cmn {
enum dir_t { DOWN= -1, NONE= 0, UP= 1 };
packet_t ptype_; // packet type (see above)
int size_; // simulated packet size
int uid_; // unique id
int error_; // error flag
int errbitcnt_; //
int fecsize_;
```

```
double ts_; // timestamp: for q-delay measurement
int iface_; // receiving interface (label)
dir_t direction_; // direction: 0=none, 1=up, -1=down
// source routing
char src_rt_valid;
double ts_arr_; // Required by Marker of JOBS
//add the following three lines
int frametype_; // frame type for MPEG video transmission
double sendtime_; // send time
unsigned long int frame_pkt_id_;
```

Step 2: You can modify the file agent.h in the common folder as follows:

```
class Agent : public Connector {
public:
Agent(packet_t pktType);
virtual ~Agent();
void recv(Packet*, Handler*);
..
inline packet_t get_pkttype() { return type_; }
// add the following two lines
inline void set_frametype(int type) { frametype_ = type; }
inline void set_prio(int prio) { prio_ = prio; }

protected:
int command(int argc, const char*const*argv);
..
int defttl_; // default ttl for outgoing pkts
// add the following line
int frametype_; // frame type for MPEG video transmission
..
private:
void flushAVar(TracedVar *v);
};
```

Step 3: To modify the file agent.cc in the common folder, write the following code:

```
Agent::Agent(packet_t pkttype) :
size_(0), type_(pkttype), frametype_(0),
channel_(0), traceName_(NULL),
oldValueList_(NULL), app_(0), et_(0)
{
}
..
Agent::initpkt(Packet* p) const
{
hdr_cmn* ch = hdr_cmn::access(p);
ch->uid() = uidcnt_++;
ch->ptype() = type_;
ch->size() = size_;
ch->timestamp() = Scheduler::instance().clock();
ch->iface() = UNKN_IFACE.value(); // from packet.h (agent is local)
ch->direction() = hdr_cmn::NONE;

ch->error() = 0; /* pkt not corrupt to start with */
// add the following line
ch->frametype_= frametype_;
...
```

Step 4: Copy the myevalvid folder (which contains myevalvid.cc, myudp.cc, myudp.h, myevalvid_sink.cc and myevalvid_sink.h) into ns2.35; for example, ns-allinone-2.35/ns-2.35/myevalvid.

Step 5: To modify ns-allinone-2.35/ns-2.35/tcl/lib/ns-default.tcl, add the following two lines

```
Agent/myUDP set packetSize_ 1000
Tracefile set debug_ 0
```

Step 6: To modify ns-allinone-2.35/ns-2.35/Makefile.in, put myevalvid/myudp.o, myevalvid/myevalvid_sink.o and myevalvid/myevalvid.o in the OBJ_CC list.

Step 7: Recompile ns-2 as follows:

```
./configure ; make clean ; make
```

If above command fails due to tcl error, try "./install in ns-allinone-2.35/"

2. Formulating video traces from raw video and simulation

Assume that EvalVid and ns-2 are installed on Ubuntu. Similar steps will work on other Linux distros too.
Download a raw yuv video sequence (you can use known yuv videos from http://trace.eas.asu.edu/yuv/). Here is an example where md_cif.yuv is used as raw video.

Step 1: Encode the raw yuv file to m4v as follows:

```
$ffmpeg -s cif -r 30 -i md_cif.yuv -c:v mpeg4 -b:v 64k -bt 32k -g 30 -bf 2
md.m4v
```

```
ab@ab-lp:~/Documents/hs$ ffmpeg -s cif -r 30 -i md_cif.yu
ffmpeg version 2.8.8-0ubuntu0.16.04.1 Copyright (c) 2000-
  built with gcc 5.4.0 (Ubuntu 5.4.0-6ubuntu1~16.04.2) 20
  configuration: --prefix=/usr --extra-version=0ubuntu0.1
--incdir=/usr/include/x86_64-linux-gnu --cc=cc --cxx=g++
le-decoder=libschroedinger --enable-avresample --enable-a
bbs2b --enable-libcaca --enable-libcdio --enable-libflite
ble-libgsm --enable-libmodplug --enable-libmp3lame --enab
inger --enable-libshine --enable-libsnappy --enable-libsc
ibvorbis --enable-libvpx --enable-libwavpack --enable-lib
l --enable-x11grab --enable-libdc1394 --enable-libiec6188
  libavutil      54. 31.100 / 54. 31.100
  libavcodec     56. 60.100 / 56. 60.100
  libavformat    56. 40.101 / 56. 40.101
  libavdevice    56.  4.100 / 56.  4.100
  libavfilter     5. 40.101 /  5. 40.101
  libavresample   2.  1.  0 /  2.  1.  0
  libswscale      3.  1.101 /  3.  1.101
  libswresample   1.  2.101 /  1.  2.101
  libpostproc    53.  3.100 / 53.  3.100
[rawvideo @ 0x1f583e0] Estimating duration from bitrate,
Input #0, rawvideo, from 'md_cif.yuv':
  Duration: 00:00:12.00, start: 0.000000, bitrate: 30412
    Stream #0:0: Video: rawvideo (I420 / 0x30323449), yuv
Output #0, ipod, to 'md.m4v':
  Metadata:
    encoder         : Lavf56.40.101
    Stream #0:0: Video: mpeg4 (mp4v / 0x7634706D), yuv420
    Metadata:
      encoder         : Lavc56.60.100 mpeg4
Stream mapping:
  Stream #0:0 -> #0:0 (rawvideo (native) -> mpeg4 (native
Press [q] to stop, [?] for help
frame=  162 fps=0.0 q=38.8 size=      43kB time=00:00:05.
trate=  69.5kbits/s
video:81kB audio:0kB subtitle:0kB other streams:0kB globa
ab@ab-lp:~/Documents/hs$
```

Step 2: Convert the m4v file to mp4:

```
$MP4Box -hint -mtu 1024 -fps 30 -add md_cif.m4v md_cif.mp4
```

```
ab@ab-lp:~/Documents/hs$ MP4Box -hint -mtu 1024 -fps 30 -add md_cif.m4v md_cif.mp4
IsoMedia import md_cif.m4v - track ID 1 - Video (size 352 x 288)
Hinting file with Path-MTU 1024 Bytes
Hinting track ID 1 - Type "mp4v:mp4v" (mpeg4-generic) - BW 66 kbps
Hinting track ID 2 - Type "mp4v:mp4v" (mpeg4-generic) - BW 66 kbps
Saving md_cif.mp4: 0.500 secs Interleaving
ab@ab-lp:~/Documents/hs$
```

As an option, you can create a reference yuv file. This video can be used for encoding loss estimation:

```
$ffmpeg -i md_cif.mp4 md_cif_ref.yuv
```

```
ab@ab-lp:~/Documents/hs$ ffmpeg -i md_cif.mp4 md_cif_ref.yuv
ffmpeg version 2.8.8-0ubuntu0.16.04.1 Copyright (c) 2000-2016
  built with gcc 5.4.0 (Ubuntu 5.4.0-6ubuntu1~16.04.2) 201606(
  configuration: --prefix=/usr --extra-version=0ubuntu0.16.04.
--incdir=/usr/include/x86_64-linux-gnu --cc=cc --cxx=g++ --ena
le-decoder=libschroedinger --enable-avresample --enable-avisyn
bbs2b --enable-libcaca --enable-libcdio --enable-libflite --en
ble-libgsm --enable-libmodplug --enable-libmp3lame --enable-li
inger --enable-libshine --enable-libsnappy --enable-libsoxr --
ibvorbis --enable-libvpx --enable-libwavpack --enable-libwebp
l --enable-x11grab --enable-libdc1394 --enable-libiec61883 --e
  libavutil      54. 31.100 / 54. 31.100
  libavcodec     56. 60.100 / 56. 60.100
  libavformat    56. 40.101 / 56. 40.101
  libavdevice    56.  4.100 / 56.  4.100
  libavfilter     5. 40.101 /  5. 40.101
  libavresample   2.  1.  0 /  2.  1.  0
  libswscale      3.  1.101 /  3.  1.101
  libswresample   1.  2.101 /  1.  2.101
  libpostproc    53.  3.100 / 53.  3.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'md_cif.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 1
    compatible_brands: isom
    creation_time   : 2016-11-23 14:11:03
  Duration: 00:00:10.00, start: 0.000000, bitrate: 187 kb/s
    Stream #0:0(und): Video: mpeg4 (Advanced Simple Profile) (
 15360 tbn, 30 tbc (default)
    Metadata:
      handler_name    : VideoHandler
    Stream #0:1(und): Data: none (rtp  / 0x20707472), 14 kb/s
    Metadata:
      creation_time   : 2016-11-23 14:11:03
      handler_name    : GPAC ISO Hint Handler
```

Step 3: Send an mp4 file per RTP/UDP to a specified destination host. The output of the mp4trace will be needed later, so it should be redirected to a file. The Source_Video_Trace file is the video trace file which has the information about each frame of the video and will be used by ns-2 as a traffic source.

```
$mp4trace -f -s 224.1.2.3 12346 md_cif2.mp4 > Source_Video_Trace
```

```
ab@ab-lp:~/Documents/hs$ ./mp4trace -f -s 224.1.2.3 12346 md_cif.mp4 > Source_Video_Trace
Track 1: Video (MPEG-4)  - 352x288 pixel, 300 samples, 00:00:10.000
Track 2: Hint (RTP) for track 1  - 300 samples, 00:00:10.000
Track 3: Video (MPEG-4)  - 352x288 pixel, 300 samples, 00:00:10.000
Track 4: Hint (RTP) for track 1  - 300 samples, 00:00:10.000
Track 5: Hint (RTP) for track 3  - 300 samples, 00:00:10.000
ab@ab-lp:~/Documents/hs$
```

Step 4: Create a network topology representing the network to be simulated in Evalvid.tcl and run the simulation as follows:

```
$ns Evalvid.tcl
```

```
ab@ab-lp:~/Documents/hs$ ns Evalvid.tcl
9.9339999999999993
300 records
```

Step 5: After simulation, ns-2 will create two files, Send_time_file and Recv_time_file (the filename as used in the tcl file), which record the sending time and received time of each packet, respectively.

Step 6: To reconstruct the transmitted video as it is seen by the receiver, the video and trace files are processed by etmp4 (Evaluate Traces of MP4-file transmission):

```
$etmp4 -f -0 -c Send_time_file Recv_time_file Source_Video_Trace md_cif.mp4
md_cif_recv.mp4
```

```
ab@ab-lp:~/Documents/hs$ ./etmp4 -f -0 -c Send_time_file Recv_time_
*** loss_md_cif_recv.txt ***: percentage of lost [frames|packets]
    column 1: I (including H)
    column 2: P
    column 3: B
    column 4: overall

*** delay_md_cif_recv.txt ***: jitter/delay statistics
    column 1: [frame|packet] id
    column 2: loss flag
    column 3: end-to-end delay [s]
    column 4: sender inter [frame|packet] lag [s]
    column 5: receiver inter [frame|packet] lag [s]
    column 6: cumulative jitter [s] [Hartanto et. al.]

*** rate_s_md_cif_recv.txt ***: sender rate
    column 1: time [s]
    column 2: momentary rate [bytes/s]
    column 3: cumulative rate [bytes/s]

*** rate_r_md_cif_recv.txt ***: receiver rate
    column 1: time [s]
    column 2: momentary rate [bytes/s]
```

This generates a (possibly corrupted) video file, in which all frames that got lost or were corrupted are deleted from the original video track. The reconstructed video will also demonstrate the effect of jitter introduced during packet transmission.
etmp4 also creates some other files, which are listed below:

loss_md_cif_recv.txt contains I, P, B and overall frame loss in percentage.
delay_md_cif_recv.txt contains frame-nr., lost-flag, end-to-end delay, inter-frame gap sender, inter-frame gap receiver, and cumulative jitter in seconds.
rate_s_md_cif_recv.txt contains time, bytes per second (current time interval), and bytes per second (cumulative) measured at the sender and receiver.

Step 7: Now, decode the received video to yuv format:

```
$ffmpeg -i md_cif_recv.mp4 md_cif_recv.yuv
```

```
ab@ab-lp:~/Documents/hs$ ffmpeg -i md_cif_recv.mp4 md_cif_recv.yuv
ffmpeg version 2.8.8-0ubuntu0.16.04.1 Copyright (c) 2000-2016 the
  built with gcc 5.4.0 (Ubuntu 5.4.0-6ubuntu1~16.04.2) 20160609
  configuration: --prefix=/usr --extra-version=0ubuntu0.16.04.1 --
--incdir=/usr/include/x86_64-linux-gnu --cc=cc --cxx=g++ --enable-
le-decoder=libschroedinger --enable-avresample --enable-avisynth -
bbs2b --enable-libcaca --enable-libcdio --enable-libflite --enable
ble-libgsm --enable-libmodplug --enable-libmp3lame --enable-libope
inger --enable-libshine --enable-libsnappy --enable-libsoxr --enab
ibvorbis --enable-libvpx --enable-libwavpack --enable-libwebp --en
l --enable-x11grab --enable-libdc1394 --enable-libiec61883 --enabl
  libavutil      54. 31.100 / 54. 31.100
  libavcodec     56. 60.100 / 56. 60.100
  libavformat    56. 40.101 / 56. 40.101
  libavdevice    56.  4.100 / 56.  4.100
  libavfilter     5. 40.101 /  5. 40.101
  libavresample   2.  1.  0 /  2.  1.  0
  libswscale      3.  1.101 /  3.  1.101
  libswresample   1.  2.101 /  1.  2.101
  libpostproc    53.  3.100 / 53.  3.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'md_cif_recv.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 1
    compatible_brands: isom
    creation_time   : 2016-11-22 19:34:38
  Duration: 00:00:10.00, start: 0.033333, bitrate: 69 kb/s
    Stream #0:0(und): Video: mpeg4 (Advanced Simple Profile) (mp4v
 15360 tbn, 30 tbc (default)
```

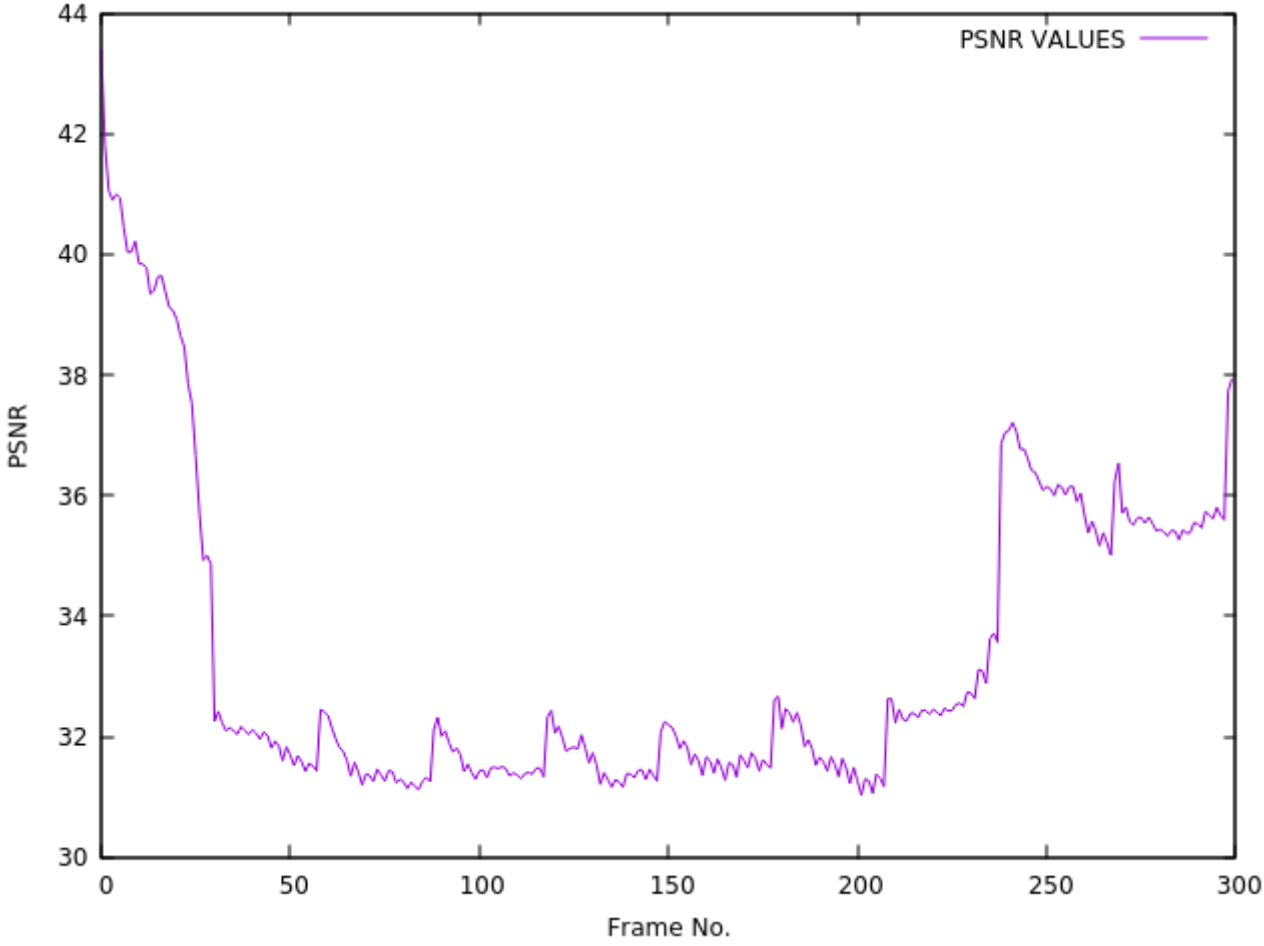
Step 8: Compute the PSNR:

$psnr 352 288 420 md_cif2.yuv md_cif_recv.yuv > ref_psnr.txt

```
ab@ab-lp:~/Documents/hs$ ./psnr 352 288 420 md_cif.yuv
psnr:   300 frames (CPU: 0 s) mean: 33.38 stdv: 2.66
```

Step 9: Plot PSNR values:

```
$gnuplot
gnuplot> set xlabel "Frame No."
gnuplot> set ylabel "PSNR"
gnuplot> plot "ref_psnr.txt" title "PSNR VALUES" with lines
```


PSNR VALUES
PSNR
Frame No.
44
42
40
38
36
34
32
30
0
50
100
150
200
250
300