NS2 Application for Video traffic

Saba Ahsan

January 6, 2016

1 Introduction

This document describes the Video traffic Application (VmApp) for Ns-2 in brief. It includes the instructions on how to compile the code and a few examples to show its usage.

2 What it does

VmApp generates video traffic using a media file and a target file. Both files consist of media bitrate values for every second. The media files are included in the git package in the folder ns2-videosrc/samplefiles/. Each media file has 7 columns (0 - 6), but VmApp only uses three of them: column 2 is the number of bytes, column 3 is the instantaneous bit rate in kbps and column 6 is the timestamp in milliseconds. The cumulative bit rate of the sample stream is calculated in the beginning by summing all the values in column 2 and dividing by the last timestamp in column 6. This is then used for calculating the sending bit rate $\mathrm{R}_{\mathrm{achieve}}$ as follows

$$R_{achieve} = \frac{R_{media}}{R_{cumulative}} \times R_{target} \tag{1}$$

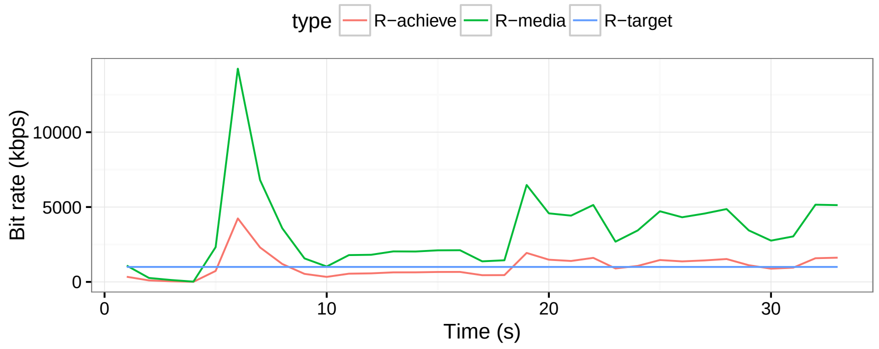

Figure 1: A comparison of the media and achieved rate when 1000kbps target rate was used.

This value is calculated per second, reading R_{media} and R_{target} from the respective files. The frame rate is defined by the user (if not the default value is used). Based on $R_{achieve}$ and the frame rate, VmApp can then calculate the size of each frame (all frames are of the same size during that second).

3 How to build

1. Place the ”vdo” directory (containing vm-udp.cc, vm-udp.h, vm-app.cc and vm-app.h) in the ns-2.35 folder[1].

2. In “common/packet.h” define PT_Multimedia and change PT_NTYPE to 74. Furthermore, add PT_Multimedia to p_info class.

```
        // insert new packet types here

static packet_t       PT_Multimedia = 73; //The new PT for MM
static packet_t       PT_NTYPE = 74; // This MUST be the LAST one
```

”packet.h”

```
        static bool data_packet(packet_t type) {
                return ( (type) == PT_TCP || \
                         (type) == PT_TELNET || \
                         (type) == PT_CBR || \
                         (type) == PT_AUDIO || \
                         (type) == PT_VIDEO || \
                         (type) == PT_ACK || \
                         (type) == PT_SCTP || \
                         (type) == PT_SCTP_APP1 || \
                         (type) == PT_Multimedia || \
                         (type) == PT_HDLC \
                       );
        }
        static packetClass classify(packet_t type) {
                if (type == PT_DSR ||
                    type == PT_MESSAGE ||
                    type == PT_TORA ||
                    type == PT_PUMA ||
                    type == PT_AODV ||
                    type == PT_MDART)
                        return ROUTING;
                if (type == PT_TCP ||
                    type == PT_TELNET ||
                    type == PT_CBR ||
                    type == PT_AUDIO ||
                    type == PT_VIDEO ||
                    type == PT_ACK ||
                    type == PT_SCTP ||
                    type == PT_SCTP_APP1 ||
```

[1]http://www.isi.edu/nsnam/ns/ns-build.html

```
                                        type == PT_Multimedia ||
                                        type == PT_HDLC)
                                                return DATApkt;
```

"packet.h"

3. In “tcl/lib/ns-packet.tcl” register the new application header

```
# Other:
        Encap      # common/encap.cc
        IPinIP     # IP encapsulation
        HDLC       # High Level Data Link Control
        Multimedia #Self define MM
```

"ns-packet.tcl"

4. In “Agent” class in “common/agent.h” add supportVM() and enableVM()

```
        inline packet_t get_pkttype() { return type_; }

        //add supportvm MM
        virtual int supportVM() { return 0; }
        virtual void enableVM() {}
```

"agent.h"

5. In “Application” class in “apps/app.h” add recv_msg() method

```
class Agent;

class Application : public Process {
public:
        Application();
        virtual void send(int nbytes);
        virtual void recv(int nbytes);
        virtual void resume();

        //vmap recv_msg MM
        virtual void recv_msg(int nbytes, const char * msg=0){};
```

"app.h"

6. Add default values for new parameters at the end of “tcl/lib/ns-default.tcl”.

```

Application/VmApp set pktsize_ 1000
Application/VmApp set target_rate_ 1000
```

"ns-default.tcl"

7. In “Makefile.in” add “vm-app.o” and “vm-upd.o” and run “./configure”/. Run “Make clean ” and then re-compile NS type “make”.

```
        wpan/p802_15_4trace.o wpan/p802_15_4transac.o \
        apps/pbc.o \
        vdo/vm-app.o \
        vdo/vm-udp.o \
        $(OBJ_STL)
```

"Makefile"

4 Examples