My Data: A Framework for Transfer of User Data Using NS2 Simulator

Masood Alam Abbasi
Faculty of Computer Science Department
Institute of Business Administration
Karachi, Pakistan
maabbasi@iba.edu.pk

Dr. Sayeed Ghani
Faculty of Computer Science Department
Associate Dean, Institute of Business Administration
Karachi, Pakistan
sghani@iba.edu.pk

***Abstract*—ns2 Simulator has been widely used to simulate various scenarios related to wired and wireless medium. However, transfer of real time user data has not been implemented as researchers are more concerned with the output of the system in terms of range of values of particular parameter desired or else. This framework besides running the default simulation as done by ns2 adds the capability to allow transfer of user data and allows user to save at the other end or destination the data received. The user data may be multimedia data or any data (plain text), document file, binary file or raw data. Simulations result show that actual user data received may be same or different, due packet error or noise, respectively depending upon the network topologies used. The user data transfer has been implemented in UDP (CBR traffic).**

I. Introduction

The transfer of multimedia data over Wireless Sensor Networks (WSNs) has always been a special requirement and is typically suited for the purpose of surveillance and remote monitoring. ns2 [1] has been widely used in the academia and industry to study the network behavior in Wired and wireless medium. Most of the researchers have been found to be focused in the evaluation of the particular protocol and its derivative and no effort has been made to transfer user data using wired or wireless medium using ns2. The aim of this research is the development of a Framework for real time user data transfer using ns2 simulator and ability to save received user data, i.e. Text/binary/image/video stream data etc.., as received.

The ns2 source code that performs low level simulation has been written in C++. Some C++ classes customized to transfer user data with existing source code have also been implemented. Similarly as with existing simulation files of NS2 (tcl files), user can pass values, as arguments, to the modified C++ class for user data file and choice of media (wired or wireless) depending upon the network configuration. The NS2 source code version ns-2.35 [2] has been used for Framework implementation.

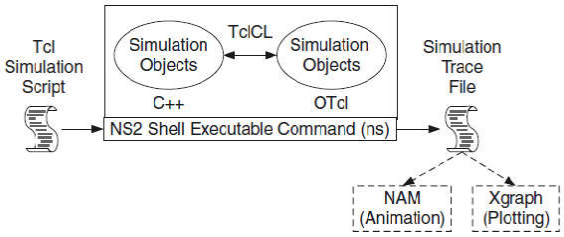

Fig. 1 NS2 Basic Architecture [3]

II. Literature Review

Implementation of Framework on ns2 Simulator platform involves study of low level classes that has been provided as part of ns2 source code [2].

Network Simulator (Version 2), known as ns2, is an event-driven simulation tool that has been used for simulating and studying the dynamic nature of communication networks. Simulation of wired/wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be easily done using the ns2 Simulator. In general, ns2 provides an interface the user can use for specifying particular network protocol and simulate its corresponding behavior. Due to flexibility of use and modular nature ns2 has been widely used in the academia and industry for network research and protocol validation. In [4], the ns2 simulator has been used for performance analysis of VANET. In [5] ns2 simulator has been used for performance analysis of multimedia traffic in MPLS network.

The basic architecture of ns2 is shown in Fig. 1 from [3]. The TCL simulation script is provided as an input to ns2 command line tool (ns). A user can generate trace file by specifying requirement in the script. The trace file provides user ability to plot graph and create animation. The ns2 simulator consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). The C++ programming language serves as heart of the simulator and act as a backbone. The simulation setup is created through the OTcl script that assembles and configures the discrete scheduling events and simulator objects. The C++ and the OTcl classes are linked together using C++ and TCL mapping classes. Moreover, for advance users or for modification as done in this research the existing C++ Classes needs to be modified or new Classes created to provide required

978-1-5386-5892-5/18/$31.00 ©2018 IEEE
ICIS 2018, June 6-8, 2018, Singapore

functionality. The ns2 simulator has capability to output either text-based trace file or graphical results (.nam) files that can be interpreted graphically and interactively using NAM (Network AniMator) tool [3].

In [6], a tool, "MyEvalvid-NT" has been developed to allow transfer of MPEG video that can be stored at the receiving node for post analysis. In [7], the details for integration of the developed tool and scenario for video transfer has been described in detail. However, only MPEG video data transfer is simulated in MyEvalvid-NT tool. It is pertinent to mention that in [6] no MPEG file is transferred as user data within the network instead the packet loss from trace file is used as input to another software developed in [6] & [7] that subtracts relevant packet from the MPEG files as if they would have lost during transmission phase, whereas in this implementation actual user data (file), in any form i.e. text, binary, image and video has been provisioned to be transferred and received that shows real values of the data, i.e. file contents.

The core of the NS2 simulator are agents that are responsible for sending/receiving data to/from lower layers. Table I below displays the UDP Agent Class structure.

Table I: UDP Agent Class from [1]

```
#ifndef ns_udp_h
#define ns_udp_h
#include "agent.h"
#include "trafgen.h"
#include "packet.h"

//"rtp timestamp" needs the samplerate
#define SAMPLERATE 8000
#define RTP_M 0x0080 // marker for significant
events

class UdpAgent : public Agent {
public:
UdpAgent();
UdpAgent(packet_t);
virtual void sendmsg(int nbytes, const char *flags = 0)
{
sendmsg(nbytes, NULL, flags);
}
virtual void sendmsg(int nbytes, AppData* data, const
char *flags = 0);
virtual void recv(Packet* pkt, Handler*);
virtual int command (int argc, const char*const*
argv);
protected:
        int seqno_;
```

The user data (text/multimedia/binary) is sent from C++ Classes implemented through Agent Class to the destination. In NS2, agents are responsible for sending data from one node to another and similarly agents receive the data and transfer data to the layers above. The scheduling of next packet (ns2 default) is also done in this class. To restore the original functionality of the NS2 classes, the classes have been rewritten and replicated where necessary preserving the original behavior of NS2 simulator.

III. IMPLEMENTATION

The implementation of the framework has been achieved by creation of various C++ classes to handle user data. Following C++ classes have been created:

a. Media_Traffic
b. MyData

The implementation of Framework has been done without affecting the existing code and examples included in the ns2-2.35 source code. The details of each of the classes implemented/ modified are as follows:

A. Class Media_Traffic

This class is responsible for assigning various transmission related parameters like:

a. Path to user data file
b. Packet transmission interval.
c. Packet Transmission rate.

The user can provide the attributes mentioned above in the corresponding tcl file and these are then passed to object of Media_Traffic Class. This class initiates data transfer, similarly the received data is passed from object of UDPAgent Class to this class object at the destination. Table III lists the Media_Traffic Class that has been implemented consisting of sending and receiving routines/functions.

Table III: Media_Traffic Class

```
class Media_Traffic : public TrafficGenerator {
 public:
Media_Traffic();
virtual double next_interval(int&);
inline double interval() { return (interval_); }
void recv(int);
void receivePacket(char*, int ,int );
virtual void send(int);
virtual int command(int argc, const char*const* argv);
```

The sending and receiving functions **void send(int)** and **receivePacket(char*, int ,int)** have been implemented to handle user data transfer. The C++ to TCL mapping has been done in the lines of code as shown in Table IV and V respectively.

Table IV: C++ to TCL Bindings for MasmediaAgent Class

```
static class MasmediaAgentClass : public TclClass {
public:
MasmediaAgentClass() :TclClass("Agent/UDP/Masmedia")
{}
TclObject* create(int, const char*const*) {
return (new MasmediaAgent());}
}
 class_masmediaudp_agent;  //class_udp_agent;
```

Table V: C++ to TCL Binding Class

```
static class MYMedia_TrafficClass : public TclClass {
 public:
MYMedia_TrafficClass() :
TclClass("Application/Traffic/CBR/Mymediaapp") {}
TclObject* create(int, const char*const*) {
return (new Media_Traffic() );
}
} class_Media_Traffic;
```

B. Class MyData

This class has been implemented to define the structure of the packet and for the purpose of adding user data in the transmission stream. Table VI lists the structure of MyData Class.

Table VI: MyData Class

```
class MyData : public AppData {

public:

  MyData(int) ;
  MyData(MyData&);
  virtual ~MyData() ;

  unsigned char* data();
  void fillData(unsigned char* ,int );
  virtual AppData* copy() ;
  virtual int size() const { return datalen_; }
  virtual AppData* copy();

  private:

  unsigned char* data_;
  int datalen_;
};
```

IV. EXPERIMENTAL SETUP

In the experimental setup static image and text/binary files have been transferred using wired and wireless (wifi) media. For wired network two configurations have been tested. In one configuration four nodes have been created with the simulation script with Nodes 0 & 1 transmitting to the destination, i.e. Node 3. The Node 2 act as a router in this case. The diagram below shows the output in the nam tool for the topology implemented.

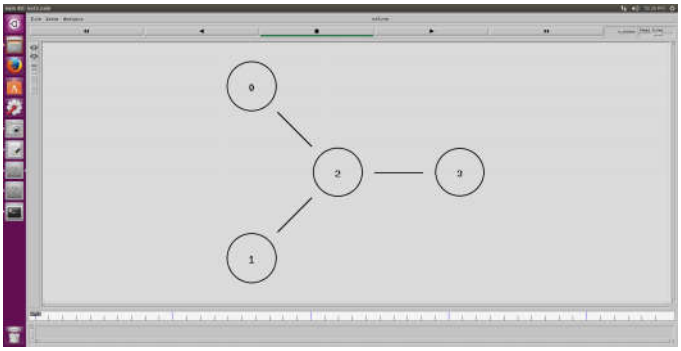

Fig.2 Simulation Topology

Nodes 0 &1 reads from the image file and binary/raw data is transferred to the destination packet by packet. The formation of user data file at the destination is based on received data according to the network topology used and it may or may not contain errors. Nodes 1 and 2 as shown in Fig 2 can be provided with the same user data file or different files and respective image file is constructed at the destination based on the transmitting nodes. Table IV below lists part of the the tcl script used to pass image path, as an argument, for transmission from developed framework. It uses UDP agent and CBR application.

The extracts of the tcl script to be used with developed framework is shown in Table VII. In the tcl script the user data (image file path) is passed to the application with syntax "send path_to_file" and at the receiving end syntax "receive path_to_file wifi or wpan" is provided to save the received data. It is pertinent to mention that multimedia files (image/video) have associated headers so for simulating the real multimedia content transfer, as in case of video streaming, no headers of the multimedia file should be transferred and one needs to extract them and after receving at the destination the same headers needs to be inserted in the received data file for display by the image/video viewing software. Any of the argument either wifi or wpan may be provided if wireless network simulation is conducted so that appropriate headers could be added /removed as necessary during data transfer, for wired network simulation no extra argument needs to be passed and " argument receive path_to_file" is sufficient.

As mentioned above, the file transferred in the tcl script show in in table VII "pixels.bmp" contains only raw image data without headers, to make raw bmp file a separate utility has been created in C++ that extracts header of bmp files and embed headers of the bmp file at the receiving end. In the

sample script shown in table VII file pixels_rcvd.bmp contains raw image/pixel data at the end of simulation and it is later on added with bmp header to made it displayable over image viewing software, in this actual image transferred and whatever received, either corrupt or error free, both are displayed.

Table VII: TCL script for wired network setup with user data

```
#This file uses UDP agent and customomized CBR Traffic
#application "Mymediaapp"
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set null1 [new Agent/Null]
$ns attach-agent $n4 $null1
$ns connect $udp0 $null1
set cbr0 [new Application/Traffic/CBR/Mymediaapp]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 50
$cbr0 set rate_ 1.0Mb
$cbr0 set random_
$cbr0 send "/home/pixels.bmp"
$ns at 0.3 "$cbr0 start"
$ns at 10.0 "$cbr0 stop"
set cbr1 [new Application/Traffic/CBR/Mymediaapp]
$cbr1 attach-agent $null1
$cbr1 set packetSize_ 50
$cbr1 set rate_ 1.0Mb
$cbr1 set random_
$cbr1 receive "/home/pixels_rcvd.bmp"
$ns at 0.1 "$cbr1 start"
$ns at 11.1 "$cbr0 stop"
```

Fig.3 shows output of the NS2 network simulator when packets are transmitted containing real data of the images selected to be transmitted from Nodes 0 and 1 to Node 3 (Sink Node). In this scenario there are two image transmitting nodes and one receiving node but node 3 will be able to save image data from node 0 and node 1 in different files respectively.

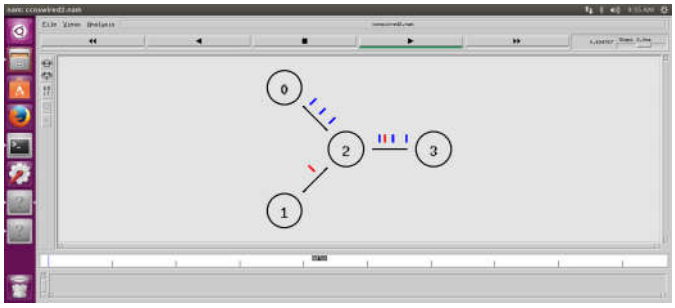

Fig.3 Nodes Transmitting Image Data to Sink

The mechanism to send data from different nodes and receive on a node using developed framework in this research is shown in table VIII for wifi network simulation.

Table VIII: TCL script for wifi-network with Tx/Rx nodes

```
set udp0 [new Agent/UDP]
$udp0 set packetSize_ 1000
$ns attach-agent $n0 $udp0
set null4 [new Agent/Null]
$ns attach-agent $n1 $null4
$ns connect $udp0 $null4
set cbr0 [new Application/Traffic/CBR/Mymediaapp]
$cbr0 set packetSize_ 300
$cbr0 set rate_ 1.0Mb
$cbr0 set random_ null
$cbr0 send "/home/aaht14/pixels.bmp"
$cbr0 attach-agent $udp0
$ns at 1.0 "$cbr0 start"
$ns at 7.0 "$cbr0 stop"
set udp2 [new Agent/UDP]
$udp2 set packetSize_ 1000
$ns attach-agent $n2 $udp2
set cbr1 [new Application/Traffic/CBR/Mymediaapp]
$cbr1 set packetSize_ 300
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
$cbr1 send "/home/aaht14/pixels.bmp"
$cbr1 attach-agent $udp2
$ns at 1.1 "$cbr1 start"
$ns at 7.0 "$cbr1 stop"
set null5 [new Agent/Null]
$ns attach-agent $n1 $null5
$ns connect $udp2 $null5
set udp3 [new Agent/UDP]
$udp3 set packetSize_ 1000
$ns attach-agent $n3 $udp3
set cbr2 [new Application/Traffic/CBR/Mymediaapp]
$cbr2 set packetSize_ 300
$cbr2 set rate_ 1.0Mb
$cbr2 set random_ null
$cbr2 send "/home/aaht14/pixels.bmp"
$cbr2 attach-agent $udp3
$ns at 1.2 "$cbr2 start"
$ns at 7.0 "$cbr2 stop"
set null6 [new Agent/Null]
$ns attach-agent $n1 $null6
$ns connect $udp3 $null6
set cbr4 [new Application/Traffic/CBR/Mymediaapp]
$cbr4 set packetSize_ 300
$cbr4 set rate_ 1.0Mb
$cbr4 set random_ null
$cbr4 receive "/home/aaht14/wifirx_4.bmp" wifi
$cbr4 attach-agent $null4
$ns at 0.12 "$cbr4 start"
$ns at 9.0 "$cbr4 stop"
set cbr5 [new Application/Traffic/CBR/Mymediaapp]
$cbr5 set packetSize_ 300
$cbr5 set rate_ 1.0Mb
$cbr5 set random_ null
$cbr5 receive "/home/aaht14/wifirx_5.bmp" wifi
$cbr5 attach-agent $null5
$ns at 0.11 "$cbr5 start"
$ns at 9.00 "$cbr5 stop"
set cbr6 [new Application/Traffic/CBR/Mymediaapp]
$cbr6 set packetSize_ 300
$cbr6 set rate_ 1.0Mb
$cbr6 set random_ null
$cbr6 receive "/home/aaht14/wifirx_6.bmp" wifi
$cbr6 attach-agent $null6
$ns at 0.1 "$cbr6 start"
```

V. PRELIMINARY RESULTS

The modifications in the NS2 source code enabled transmission of payload (user data) successfully at the destination. All the experimental network configurations tested in these experiments resulted in proper transfer of payload data from source to destination using wired and wireless media.

Fig.4 shows wifi star simulation configuration, whereas Fig.5 & 6 shows transmitted and received image file as received from different nodes at the destination respectively.

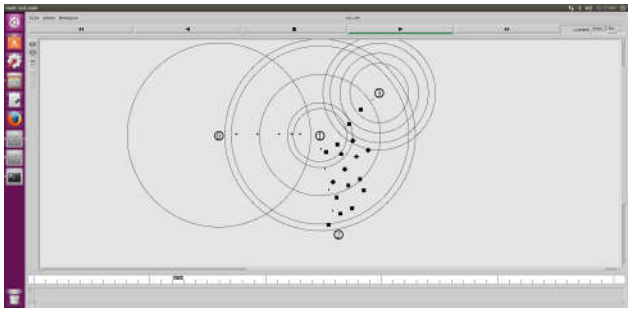

Fig.4 Wifi Star network scenario using framework

Fig.5 Transmitted bmp file

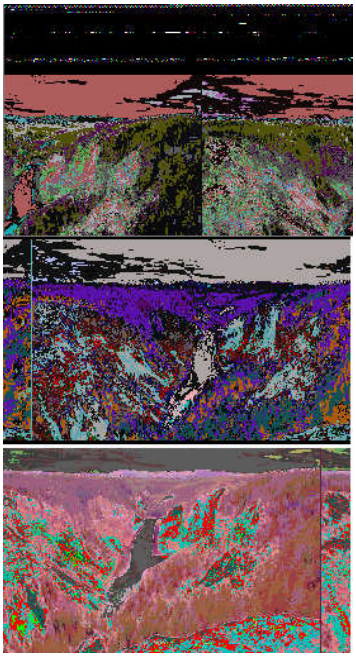

Fig.6 03 x received bmp data (header added)

Similarly transfer of user data, text file in this case, was performed by transferring text file in the network topology as shown in Fig. 4. The original contents of the text file are shown in table IX and received data/ payload from two nodes is shown in table X & X1 respectively.

Table IX: Contents of text file transmitted by 02 x node of wifi star network

The IBA launched its Bachelor programs in Business Administration in 1982, conducted under the aegis of the Karachi University, these 3-year programs continued till 1994. Upon acquiring degree awarding status that year, the nomenclature, the bachelor programs was changed to bring it in consonance with international standards.

Table X: Contents of text file received at node1 from node no. 0

The IBA launched its Bachelor programs in Business Administration in 1982, conducted under the aegis of the Karachi University, these 3-yeariring degree awardinthe nomenclature, the bachelor programs th international standards

Table XI: Contents of text file received at node 1 from node no. 2

The IBA launched its Bachelor programs in Business Administration in 1982, conducted under the aegisersity, these 3-yeartill 1994. Upon acquiring degree awarding status that year, it in consonance with international standards

Similarly several other network configurations have been tried to validate the transfer of user data. Fig. 7 shows another network configuration used in the simulation where one to one (peer configuration) data has been transmitted using two nodes only.

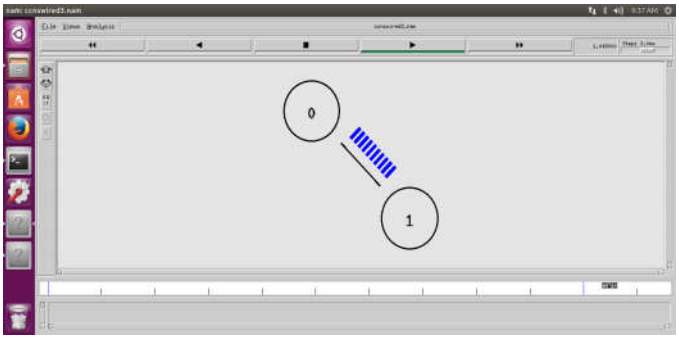

Fig.7 One to One image data transfer

VI. CONCLUSION

The modification in the NS2 source code for transfer of user data in the packet payload allows user to test data transfer using different protocols (wired or wireless). The significance of payload data transfer is not just limited to text or image files, any kind of user data like multimedia, binary or plain text can be used for transmission over the network. In the experiments conducted image and text/binary files have been transferred successfully.

The classes implemented/modified in this research proved to be useful for future research on the subject and same can be used to transfer real time video with some modifications in the Framework. Furthermore, NS2 framework developed could be further improved by transferring image data received to the browser or any standalone application for viewing the image received. The application of Framework developed is well suited for general content transfer and especially for the Multimedia research applications.

REFERENCES

[1] The Network Simulator NS2, http://www.isi.edu/nsnam/ns/. NS2 Source https://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.35/

[2] Issariyakul, Teerawat, and Ekram Hossain. *Introduction to network simulator NS2*. Springer Science & Business Media, 2011.

[3] NS2 Framework Source Code maabbas@iba.edu.pk

[4] Teddy Mantoro, Muhammad Reza "Performance Analysis of AODV and DSDV Using SUMO, MOVE and NS2".

[5] Bukashkin, Sergey, Marina Buranova, and Aleksandr Saprykin. "An analysis of multimedia traffic in the MPLS network in ns2 simulator." *Problems of Infocommunications Science and Technology (PIC S&T), 2016 Third International Scientific-Practical Conference*. IEEE, 2016.

[6] Yu, Chia-Yu, et al. "MyEvalvid-NT-A simulation tool-set for video transmission and quality evaluation." *TENCON 2006. 2006 IEEE Region 10 Conference*. IEEE, 2006.

[7] Ke, Chih-Heng, et al. "An evaluation framework for more realistic simulations of MPEG video transmission." *Journal of Information Science & Engineering* 24.2 (2008).