International Journal of Innovative Computing, Information and Control
Volume **10**, Number **1**, February **2014**

ICIC International ©2014 ISSN 1349-4198
pp. 1–18

LABNS2 – CO-SIMULATION, CO-EMULATION, AND REAL-TIME CONTROL TOOLKIT FOR INVESTIGATION OF NETWORK INDUCED TIME DELAYS AND PACKET LOSS IN NETWORKED CONTROL SYSTEMS

Ncedo Sandiso Mkondweni and Raynitchka Tzoneva

Electrical Engineering Department
Cape Peninsula University of Technology
PO Box 1906, Bellville 7535, South Africa
ncedom@gmail.com; tzonevar@cput.ac.za

Received November 2012; revised March 2013

Abstract. *In this paper, a new co-simulation, co-emulation, and real-time control toolkit for the investigation of the influence of network induced time delays and packet loss in a typical Networked Control System (NCS) is introduced. The architecture and functionality of the developed toolkit are described. The name of the toolkit is LabNS2 which is derived from co-simulation and co-emulation capabilities based on LabVIEW^{TM} and Network Simulator version 2 (NS-2). The toolkit is developed based on Windows Operating System environment; however, NS-2 is developed based on UNIX environment and as a result Cygwin software is used to emulate UNIX environment within the Windows environment. Hardware and Software parts of the toolkit are described and discussed. Finally, to demonstrate the functionality of the toolkit a DC motor control system as used in the Radio Astronomy dish telescopes and satellite systems is considered in the conditions of a communication network. The tookilt is used to measure and analyze the network induced delay. The toolkit is a simple, scalable and effective for investigation of network induced time delay and packet loss in NCSs using LabVIEW^{TM} and NS-2. The toolkit is useful for educational and research purposes and can be applied in NCSs for control of distributed plants.*

Keywords: Networked induced time delay, Packet loss, Time delay between the controller and the actuator, Time delay between the sensor and the controller

1. Introduction.

Networked control systems overlap between two areas of study that are Information Systems and Feedback Control Systems as shown in Figure 1.

The above two fields have been co-existing in isolation for decades and as a result both fields have had their own simulation and emulation tools used during design and implementation of applications used in these fields. As it is well known, Networked Control Systems are control systems where the plant and the controller exchange information via a shared communication network and the network is considered as part of the closed loop control system [1,23-26]. Unfortunately the communication network introduces random varying time delays and data packet loss.

These network imperfections degrade the performance of the closed loop control system and result in closed loop system instability [22]. The complexity of measuring the communication network imperfections in the networked control systems makes it difficult for the control engineers to develop methods for design of controllers that can incorporate and compensate these imperfections in order to improve the performance of the networked control systems [16]. The network delays and packet loss are considered to be between the controller and the actuator and between the sensor and the controller and are mostly assumed to be either constant [16] appearing in a singular or a simple form or time varying [4]. In the real environment the case of constant delays does not exist as the delays

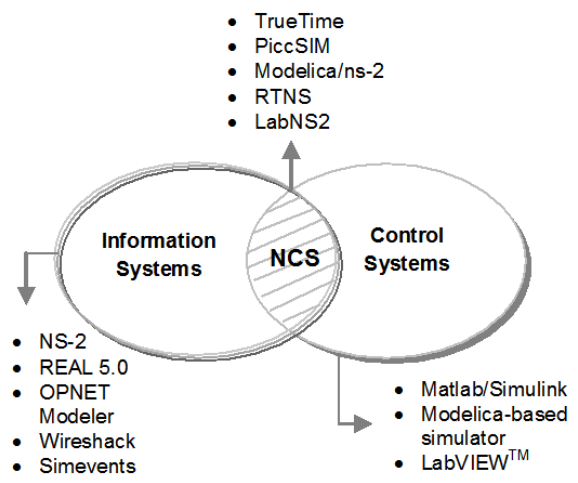


FIGURE 1. Simulators and emulators used in the information systems, control systems, and networked control systems

vary depending on the network dynamics. In some cases probabilistic Markov chain and Independent Identical Distribution (iid) Bernoulli process are used to represent the time varying delays and the packet loss [17-21]. For the case of the packet loss if the network breaks the independent Bernoulli process parameter is zero; otherwise it is one [21]. In NCS a purely analytical approach is often not possible when analysis and design of the system is performed. Instead of representing the delays in a singular form, Nilsson [17,18] modeled the delays as independent random delays and used an empirical approach to determine the model for the network induced delays using CAN bus as a protocol. These simulations are performed using Matlab and Simulink.

Due to the complexity of measuring network imperfections some control engineers have developed co-simulation tools that leverage from the existing simulators and emulators used in information systems and control systems; co-simulation tool based on Matlab/Simulink and NS-2 (PiccSIM) [3], co-simulation tool based on Modelica and NS-2 (Modellica/ns2) [9], and co-simulation tool based on RealTime and NS-2 (RealTimeNS) [11] are proposed. The choice of using NS-2 is because it is widely applied as a simulator and emulator in the information systems field [8]. Another approach used by control engineers is to develop custom-built discrete-event simulators inside Matlab/Simulink as the TrueTime [10] and NCsimulator Toolkit [16].

The drawbacks of these simulators are that they cannot be incorporated in the real-time environment, because they use only the simulation module of NS-2 and not the real-time schedule module that allows for injection of live network data into the co-emulator to measure network induced time delays and packet loss. The code developed in these simulators cannot be compiled to generate a standalone executable code for real-time implementation.

The paper describes solution of the problem for real-time measurement of the network induced delays and packet loss and the use of these measurements in generating the closed loop control. The main functions of the toolkit are:

1. Simulation and emulation of the plant and controller in the NCS;
2. Simulation and emulation of the network, implementing TCP and UDP protocols;
3. Real-time measurement of random delays.

The main contributions in this paper can be summarised as follows:

1. Based on real-time scheduler within NS-2, a real-time co-emulator, co-simulator based on the LabVIEW and NS-2 (LabNS2) is developed;
2. Based on LabVIEW, general functions called virtual instruments (vi) that represent the plant and the controller are developed;
3. Based on Tool Command Language (Tcl) and Gawk, scripts that measure in real-time the network induced time delay and packet loss are developed.

The paper presents LabNS2, a simple, scalable, and effective tool for simulation and emulation of NCS using LabVIEW^{TM}, and NS-2. The architecture of the LabNS2 toolkit is described and discussed in Section 2. The simulation and emulation engine used to realise the interface between LabVIEW and NS2 is described and discussed in Section 3. The developed Tool Command Language (Tcl) and Gawk scripts are described in Section 4. LabVIEW code that is developed is discussed in Section 5 and Section 6. Implementation of the LabNS2 for the case of a dish antenna NCS is illustrated in Section 7. Operation procedure and the results of LabNS2 are outlined in Section 8 followed by the conclusion in Section 9.

2. **Architecture of the LabNS2 Toolkit.** The architecture of the LabNS2 is based on LabVIEW, NS-2, Tcl, Gawk, and Cygwin software environments. LabVIEW^{TM} is a graphical programming language where icons representing different functions are connected to define functionality of a given system. LabVIEW functions are called Virtual Instruments (vi). In this paper LabVIEW is used to develop software modules that implement the plant, the controller, and the reference trajectory on the basis of an example of a dish antenna network control system. The software modules communicate with each other by using UDP data sockets.

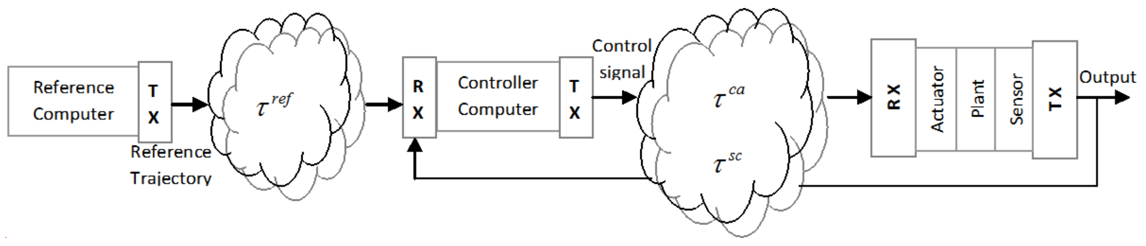
Network Simulator version 2 (NS-2) is a discrete event network simulator designed for network related research, with focus on wired and wireless networks, devices and protocols [3].

NS-2 is developed in C++ and Object Tool Command Language (OTcl) to manage, control and implement data paths in communication networks [7]. The simulator supports a class compiled hierarchy in C++ and the corresponding interpreted hierarchy within the OTcl. C++ is used for protocol implementation and for all the implementations where every packet journey has to be processed. OTcl is used for setup and configuration. NS-2 supports the following traffic protocols FTP, CBR, TCP, HTTP, Real-Time Ethernet data, and the drop tail queuing management system [2]. UDP traffic is considered encapsulated within the TCP stack to form sockets. NS-2 Emulator engine is used for the real-time Ethernet implementation. In this paper both the simulation and the emulation capabilities of NS-2 are used to analyze network dynamics in an NCS.

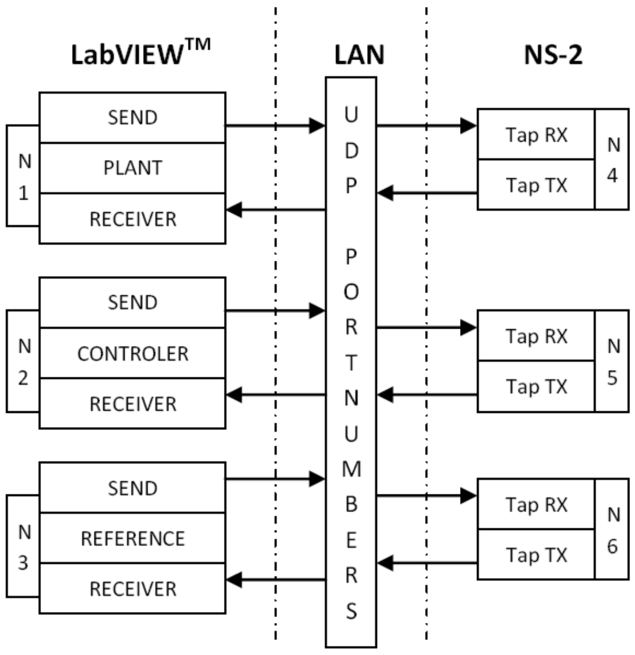
Tcl scripts are developed to access the OTcl using Tcl syntax, e.g., assigning of variables (*set a 3*), procedures (*proc sum* {*a b*}{*expr $a + $b*}). Using Tcl scripting, scripts have been developed to measure network induced time delays and packet loss.

Cygwin emulates UNIX environment within the Windows environment. The toolkit is built for a network control system as shown in Figure 2(a). The proposed architecture of the toolkit is shown in Figure 2(b). The architecture is based on a node approach where the plant, controller, and supervisory computer are considered as nodes of the closed loop control system in Figure 2(a). Each node has a built in Ethernet module which allows for sending and receiving of Ethernet datagram. Ethernet module supports UDP traffic protocol. There is scope to extend the modules to support other protocols like TCP.

To measure packet loss and time delays between the nodes the Trace Queuing Method within the NS-2 software package [8] is used where all the network events are recorded in a trace file. The trace file format is shown in Table 1. The script that allows for recording



(a)



(b)

Figure 2. (a) Network control structure. TX – send data, RX – receive data, τ^{ref} – delay between the reference computer and the controller, τ^{ca} – delay between the controller and the actuator, τ^{sc} – delay between the sensor and the controller. (b) Toolkit structure. Note that "N1" means node 1.

of the network events is written here using Tcl software. The recorded file is analyzed and processed using specially developed Gawk scripts.

Note that the measuring scripts run at the same time as the nodes; however, the Gawk scripts are executed based on the recorded output files of the Tcl scripts. LabVIEW™ and NS-2 are used to form the real-time Network Control System environment. Plant, controller and reference trajectory generating computer programs are all implemented in LabVIEW™. Network simulation and emulation is implemented using NS-2. Measurement of network induced time delay and packet loss is implemented using Tcl scripts. Processing of the recorded network event trace file is implemented in Gawk.

Nodes may be implemented on a single or multiple computers. It is recommended that each node be implemented on a separate computer.

3. **NS-2 Simulation and Emulation Engine.** The ability of the toolkit to measure live network traffic is based on the NS-2 emulation facility which exposes the simulator to the live network traffic [7]. The emulator has built in network object which allows for simulator to receive live network traffic and to inject live network traffic into the

network [12]. The network object is based on a tap interface. A tap interface is a virtual Ethernet interface that looks like network hardware to the Operating System (OS) [13]. Instead of writing bits to the Ethernet interface, the tap interface writes the bits to a user space. The network object has an associated Tap Agent class. The tap agent sends and receives packets between network objects as described in Figure 3. The tap agent handles the setting of the common header packet size field and the field type. The simulator environment can now be exposed to the network and be accessed as a node within the network. The network object allows access to the protocol layer, e.g., link layer, raw IP, and UDP by particular access mode (read-only, write-only, or read-write). LabNS2 toolkit uses this feature as a core for interfacing LabVIEW^{TM} and NS-2. The main advantage of LabNS2 is the ability to interact with the real time system, continuous or discrete and be

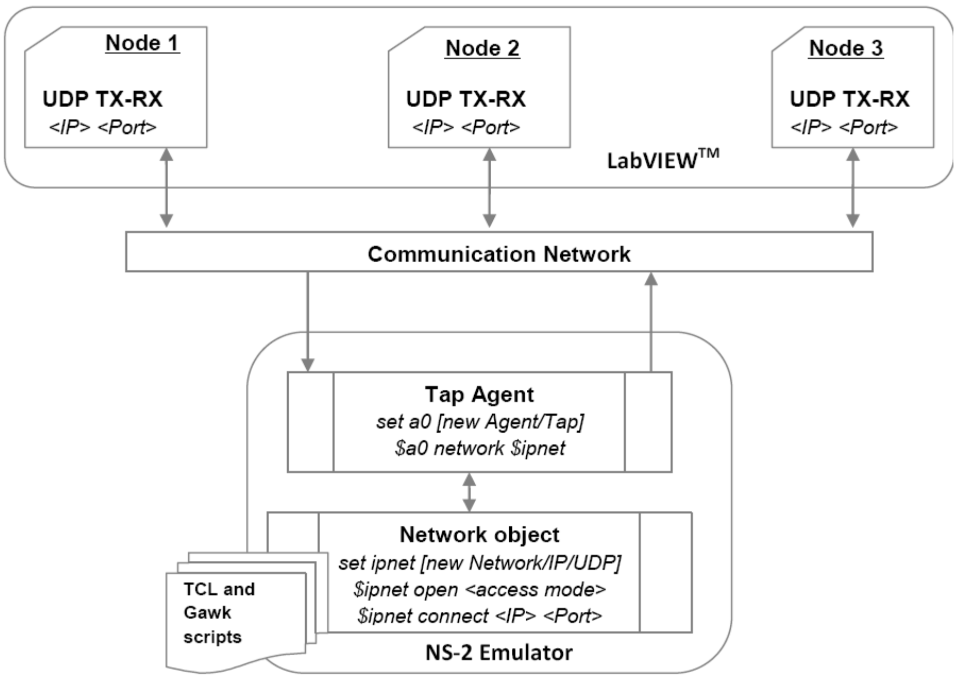


FIGURE 3. Communication structure and interface between LabVIEW^{TM} and NS-2 to make LabNS2 NS-2 emulator

TABLE 1. NS2 trace file format

Event	Abbreviation	Type	Value
Normal Event	r: Receive d: Drop e: Error +: Enqueue −: Dequeue	double	Time
		int	(Link-layer) Source Node
		int	(Link-layer) Destination Node
		string	Packet Name
		int	Packet Size
		string	Flags
		int	Flow ID
		int	(Network-layer) Source Address
		int	Source Port
		int	(Network-layer) Destination Address
		int	Destination Port
		int	Sequence Number
		int	Unique Packet ID

able to measure the network induced time delays and packet loss. The feature makes it possible to design the controller or to change the parameters of the controller in real-time based on actual measurement of the network imperfections.

4. **Tcl and Gawk Scripts Development.** The Tcl and Gawk are scripts developed to measure network induced time delays and packet losses within the NCS. The functions, output parameters and names of the scripts are shown in Table 1. Gawk scripts parse and process the trace file that is generated by NS-2 Tcl scripts during simulation and emulation. Using the scripts various modifications can be made to the network parameters, e.g., varying the channel bandwidth and channel drop tail. The developed scripts are listed in Table 2.

TABLE 2. List of the developed scripts in Tcl and Gawk software

Script	Input	Output	Function
measure-tdpl-rc-aN.tcl <cbw> <ex_time>	Channel band width (cbw) and execution time.	Recorded trace file	Measures time delay and packet loss between reference computer and the controller.
measure-tdpl-ca-aN.tcl <cbw> <ex_time>	Channel band width (cbw) and execution time.	Recorded trace file	Measures the time delay and packet loss between the controller and the actuator.
measure-tdpl-sc-aN.tcl <cbw> <ex_time>	Channel band width (cbw) and execution time.	Recorded trace file	Measures the time delay and the packet loss between the sensor and the controller.
calculate_packet _loss.awk	NS-2 Trace file	Packet sent, lost and received	Calculates packet loss from the trace file.
Calculate-time_delay.awk	NS-2 Trace file	Excel file	Calculates time delays from the trace file.
cbw = Channel Bandwidth, ex_time = Execution time			

5. **Development of the LabVIEW^{TM} Software.** LabVIEW is used to implement the plant, the controller and the reference generating trajectory computer programs. LabVIEW uses virtual instruments (vis) as functions within the graphical programming environment. These vis allow for modularity of the code as it is possible for different modules to be implemented. Table 3 shows the block library of sub-vi developed for LabNS2 and their respective functions for the case of position control of the dish control.

6. **Development of the Socket Parser.** LabVIEW software is developed for the implementation of sockets which are used to communicate with NS-2. Socket is a software endpoint that establishes bidirectional communication between a server program and one or more client programs. Sockets use standard protocol, like IP, UDP and TCP. A socket binds an IP (Internet Protocol) address and a UDP port. IP is a best effort protocol, it is connectionless, and there is no guarantee of packet delivery because IP packages the datagram and sends it without verifying that the connection exists. There is also no acknowledgement from the recipient that the packet has been delivered. The fact that there is no acknowledgment on the IP protocol brings another dimension to the Network

Table 3. List of the developed LabVIEW subvi

Node / VIs	Sub-vi Name	Input parameters	Output parameters file type	Function
Node 1 Coordinates systems and a reference trajectory	lt2ut.vi	LT (String format)	UT (Y:M:D:H:M:S)	Convert LT to UT.
	Juliandate.vi	UT (Y:M:D:H:M:S)	JD and GST	Convert UT to GST
	hms2dd.vi	Hours, Minutes, Seconds (HMS)	Decimal degrees	Convert from Hour: Minutes : Seconds format to Decimal degrees
	dms2deg.vi	Degrees, Minute, Seconds (DMS)	Decimal degrees	Convert from DMS to Decimal degrees
	Azel.vi	RA, DEC, GST, Observer Long, Observer Lat	Decimal degrees	Convert from RA, DEC to Alt Az using Ephemeris (Almanac) equations
Network Connectors	udp_tx_aN.vi	(Az, Alt) (String)	UDP datagram	Receive and send datagram via the network
	udp_parser.vi	UDP datagram (String)	Control values (double)	Parse data from udp_tx_aN.vi and sends control values to the plant.
Node 2 Controller	PID.vi	Kp, Ki, Kd	Control signal calculations (double)	Implement the PID controller.
	ss_con.vi	Matrices A, B, C and H	Control signal calculations (double)	Implement state space controller for Proportional and Proportional and Integral control.
Node 3 Plant	dish_n_tf.vi	Numerator and Denominator	Desired set point (double)	Implement the plant model in transfer function form, where n is the dish number.
	dish_n_ss.vi	Matrices A, B and C	Dish position (double)	Implement the plant model in state space form, where n is the dish number.

where LT = Local Time, UT = Universal Time, JD = Julian Date, GST = Greenwich Sidereal Time, RA = Right Ascension, DEC = Declination, Az = Azimuth, Alt = Altitude, Controller Kp = Proportional constant, Controller Ki = Integral constant, Controller Kd = Derivative constant, A, B, C are matrices of the state space model equation and measurement equations with appropriate dimensions and H is the state space controller gain feedback matrix.

Control System as the acknowledgement is critical in feedback control systems. The IP addresses and UDP port allocation for considered network control system are given in Table 4.

Table 4. UDP port allocation table

Description	**LabNS2**	**Internal Tap IP address**	**Port IN (RX)**	**Remote PC IP address**	**Port OUT (TX)**
RC – UDP TX Antenna 1	LabVIEW™	192.168.1.2	4490	192.168.1.4 *(127.0.01)*	4420
RC – UDP TX Antenna 2	LabVIEW™	N/A	4490	192.168.1.3 *(127.0.0.1)*	4400
ns-measure-rc-a1	NS-2	224.1.127.2	4420	192.168.1.4 *(127.0.0.1)*	4430
Antenna 1 (Controller + Plant)	LabVIEW™	192.168.1.4 *(127.0.0.1)*	4430	N/A	N/A
ns-measure-rc-a2	NS-2	224.1.127.2	4400	192.168.1.5 *(127.0.0.1)*	4410
Antenna 2 (Controller + Plant)	LabVIEW™	192.168.1.5 *(127.0.0.1)*	4410	N/A	N/A

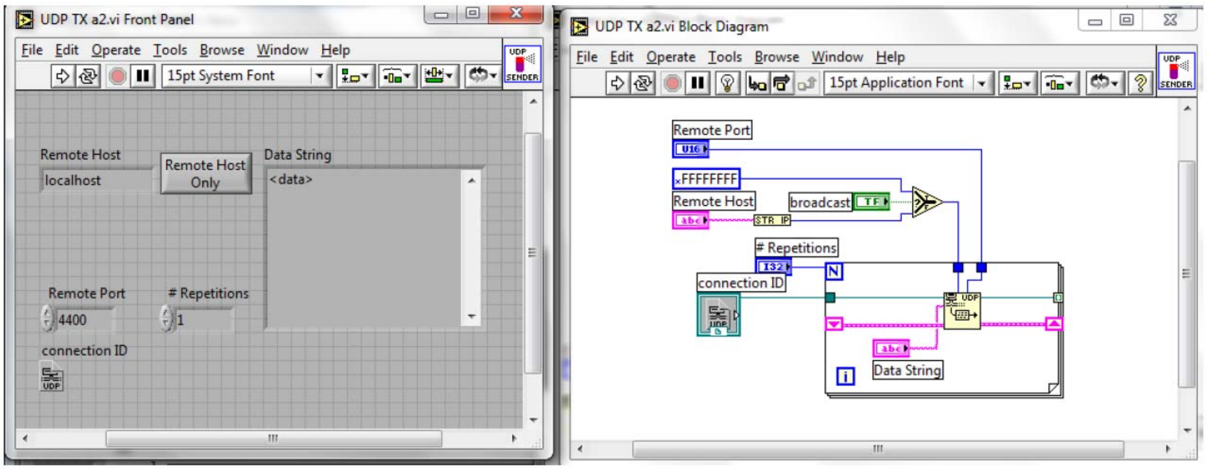


Figure 4. Front panel and block diagram subvi of sockets implementation

The glue between LabNS2 code and NS-2 is through Ethernet interface. NS-2 real-time scheduler is used to interface with real live network. It allows for sending and receiving of raw UDP data back and forth thereby creating an environment for measurement of network induced time delay and packet loss. The LabVIEW subvi that implements the sockets is shown in Figure 4.

7. **Implementation of the LabNS2 Emulator for the Case of a Dish Antenna NCS.** To illustrate how the emulator operates a typical NCS framework is used whereby a DC motor model is considered as used in Radio Telescope dishes and satellite systems. The control structure of the typical Radio Telescope dish is shown in Figure 5. This control structure has four sub-elements that are Coordinate Transformer, Model of the DC Motor (Plant), Controller, and NS-2 Emulation Engine with added the developed Tcl and Gawk scripts. Typically the reference trajectory is considered to be the object in the sky identified with the Right Ascension (RA) and Declination (DEC) coordinates. A coordinate transformer is required to convert from RA, DEC to Altitude and Azimuth [Alt, Az] coordinates. The [Alt, Az] becomes the reference point that is sent to the

respective controllers. The controllers are typically PID or Pole Placement state space ones. The emulation structure of the NCS for the case of a Radio Telescope or Satellite system for the case of Azimuth control is shown in Figure 6. Computer 1 of Figure 6 implements the reference trajectory software and the controller software developed in LabVIEW. The controller and the plant are connected via the network using NS-2 to measure the time delay and packet loss between the controller and the actuator and between the sensor and the controller as shown in Figure 6. Computer 3 in Figure 6 executes the Tcl scripts to measure time delay and packet loss. Computer 2 of Figure 6 executes the plant model, sends and receives delayed control action values, and sends sensor measurement values to the NS-2 computer that further sends the data to the controller's computer. The emulator diagram for the Altitude NCS is the same as this in Figure 6.

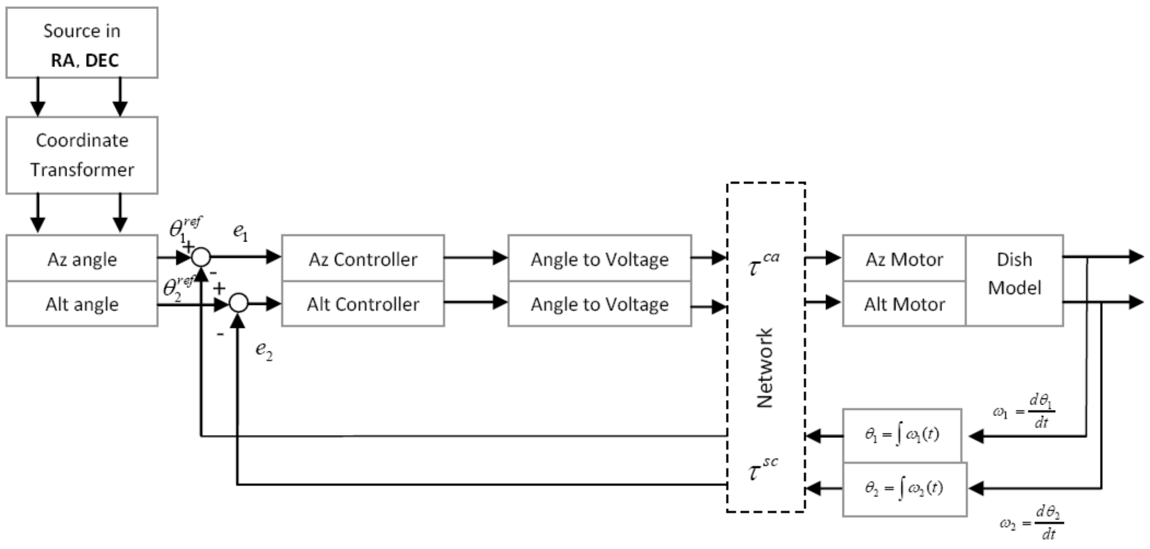


Figure 5. Control structure of a typical radio telescope

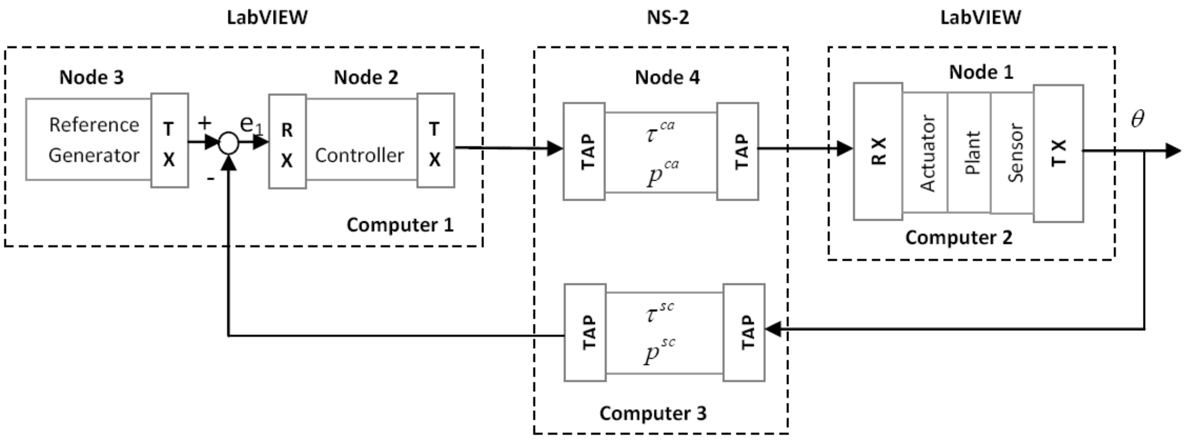


Figure 6. Emulation block diagram for the Azimuth angle NCS, where p^{sc} is the packet loss between the sensor and the controller and p^{ca} is the packet loss between the controller and the actuator

7.1. **Coordinate transformer – Reference trajectory generator.** Earth coordinates are different to the coordinates of the object being observed by the telescope. The coordinates of the object are called celestial equatorial coordinates. Similarly to earth coordinates, the analogue for the longitude is Right Ascension and the analogue for the latitude is declination in equatorial coordinates.

Right ascension of a star is the angular distance between the meridians through the first point in Ares and the considered star. Ares is a zero point chosen on the celestial sphere at which the Sun crosses the equator, on March 21. This is the northern Vernal (spring) Equinox, and the southern hemisphere autumnal Equinox.

Declination of a star is the angular distance measured in degrees between the celestial equator (0^0) and the star. The north celestial pole is at positive ninety degrees and the south celestial pole is at negative ninety degrees. Horizontal coordinates also called alt-az system are used to convert from earth coordinate system (latitude/longitude) to celestial equatorial coordinate system (right ascension/declination) using observer's local horizon as the fundamental plane.

Using equation adapted from the Ephemeris Almanac [14] the RA and DEC are converted to Alt and Az coordinates using Equations (1)÷(3).

$$h = LST - \alpha \tag{1}$$

$$Az = \arctan\left(\frac{-(\sin(h)\cos(\delta))}{\cos(\varphi)\sin(\delta) - \sin(\varphi)\cos(\delta)\cos(h)}\right) \tag{2}$$

$$Alt = \arcsin(\sin(\varphi)\sin(\delta) + \cos(\varphi)\cos(\delta)\cos(h)) \tag{3}$$

where h = Object's hour angle defined as an angular distance measured east or west of the observer's zenith meridian, α = Object's right ascension, δ = Object's declination, φ = Object's latitude, λ = Object's longitude, Az = Object's azimuth, Alt = Object's altitude, LST is the Local Sidereal Time.

The equations described above are implemented in LabVIEW using the sub-vi's described in Table 3. The altitude and the azimuth are used as reference for the position controller. Figure 7(a) and Figure 7(b) show the block diagram and the LabVIEW front panel of the developed subvi (Azel.vi) that implements Equations (1)-(3). The results are verified by using Radio Eyes [15] a commercial of the shelf software used by astronomers to look at objects in the sky.

The output of the reference trajectory is sent to the controller. The controller and the reference trajectory are implemented on computer Number 1, see Figure 6.

The LabVIEW subvi's that implement reference trajectory node is shown in Figure 8. The results are discussed in Section 8.

7.2. **Plant model.** The modeling of the two motors for azimuth and altitude is considered separately as each motor has its own set point and dynamics. The motor models are developed using Kirchhoff and Newton's laws based on the electro-mechanical structure of the DC motor as shown in Figure 9.

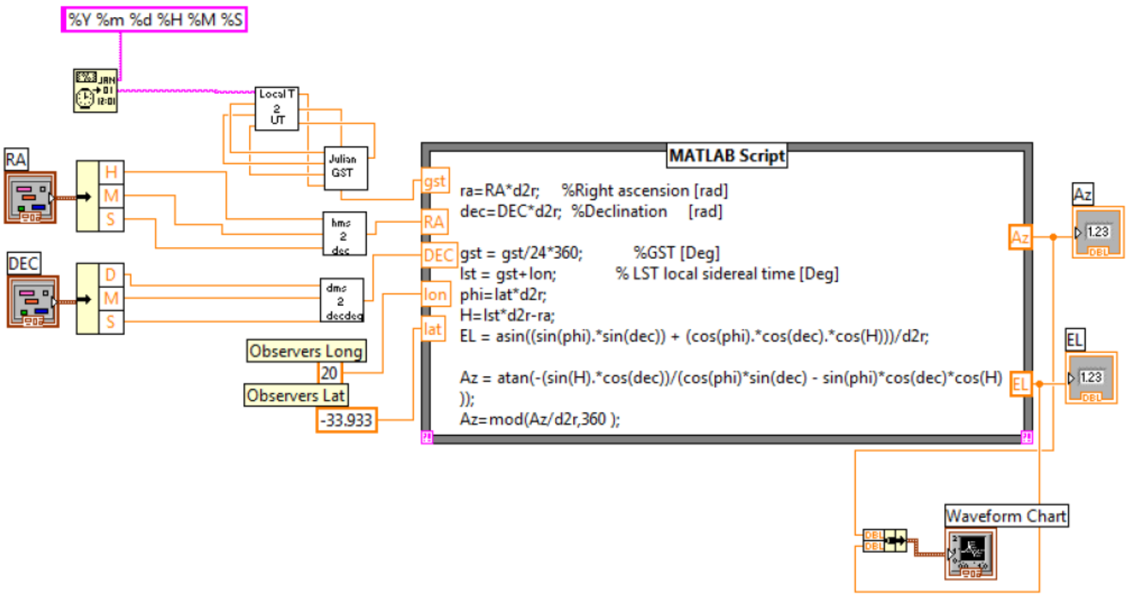
Using Kirchhoff laws and considering Figure 9,

$$V - e_a - L\frac{di}{dt} = iR \tag{4}$$

Using Newton's law and considering Figure 9,

$$T_m = J_m\dot{\omega}_m + B_m\omega_m \tag{5}$$

Considering Laplace transform of Equations (4) and (5) both equations are expressed in terms of current. Based on the two equations the transfer function of each motor is

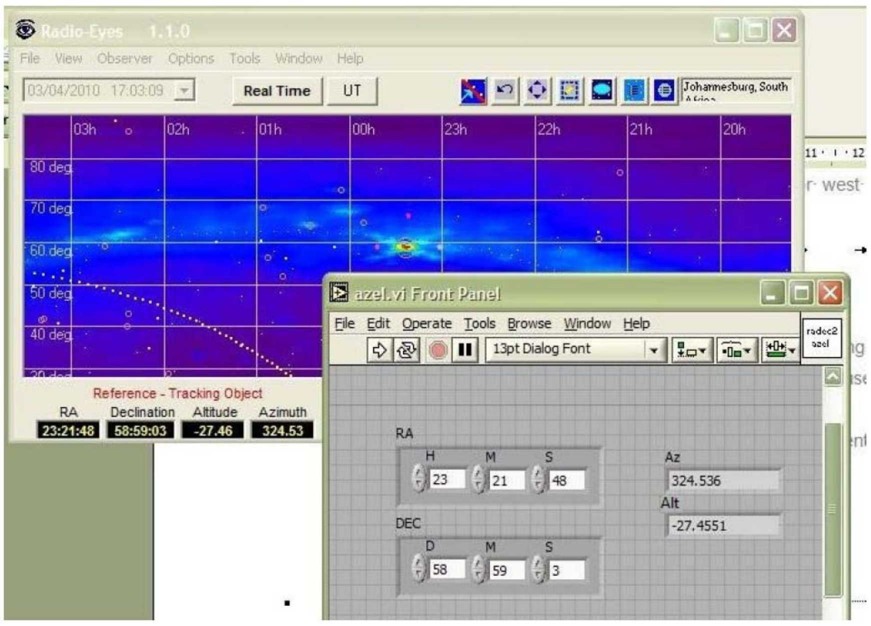


```
ra=RA*d2r;     %Right ascension [rad]
dec=DEC*d2r;  %Declination     [rad]

gst = gst/24*360;          %GST [Deg]
lst = gst+lon;           % LST local sidereal time [Deg]
phi=lat*d2r;
H=lst*d2r-ra;
EL = asin((sin(phi).*sin(dec)) + (cos(phi).*cos(dec).*cos(H)))/d2r;

Az = atan(-(sin(H).*cos(dec))/(cos(phi)*sin(dec) - sin(phi)*cos(dec)*cos(H)
));
Az=mod(Az/d2r,360 );
```

(a)



Reference - Tracking Object

RA	Declination	Altitude	Azimuth
23:21:48	58:59:03	-27.46	324.53

RA

H	M	S
23	21	48

DEC

D	M	S
58	59	3

Az 324.536

Alt -27.4551

(b)

FIGURE 7. (a) Block diagram of the right ascension declination coordinates conversion to the azimuth and altitude coordinates and (b) front panel of the right ascension declination coordinates conversion to the azimuth and altitude coordinates.

represented by Equation (6).

$$\frac{\theta_m(s)}{V(s)} = \frac{K_t/RJ_m}{s\left(s + \frac{RB_m + K_t K_b}{RJ_m}\right)} \tag{6}$$

where T_m = Toque of a motor; ω_m = angular velocity; θ_m = position of the shaft; J = Moment of inertia; R = Armature Resistance, V = Input voltage; $L \approx 0$ = Armature Inductance; B = Damping Ratio.

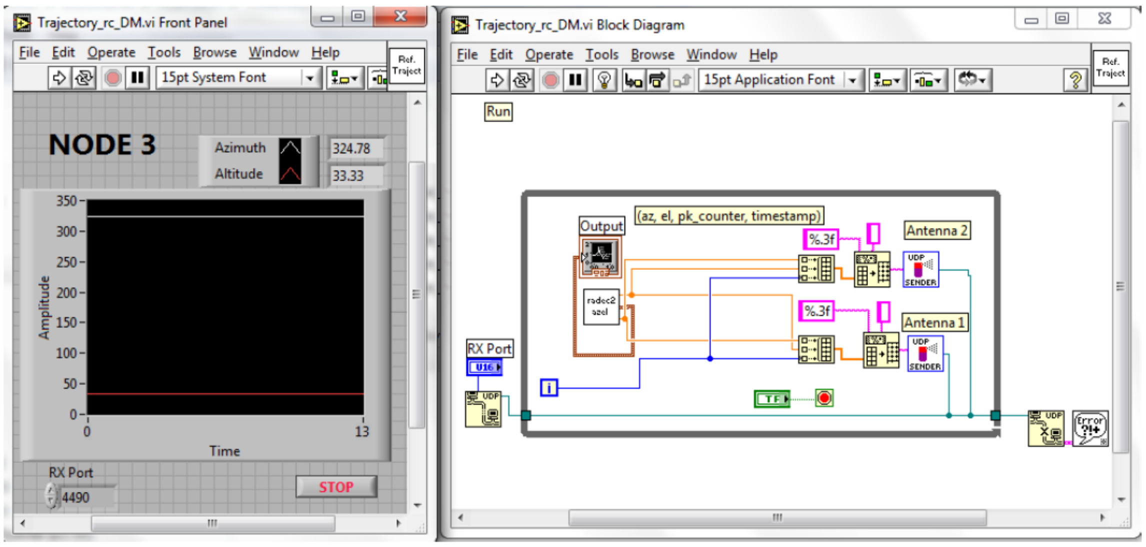


FIGURE 8. Reference vi with transmit module

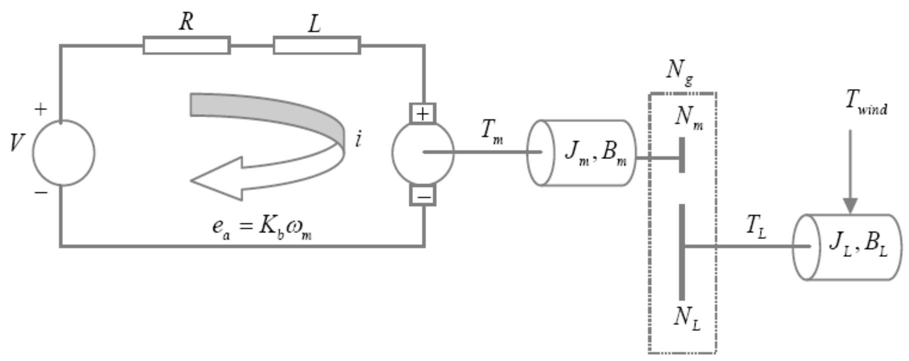


FIGURE 9. Electro-mechanical structure of the DC motor

It is assumed that K_t and K_b are equal and k^2 is used to represent the two constants. Equation (6) is transformed to state space form:

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{B_m R + k^2}{J_m R} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{K_t}{J_m R} \end{bmatrix} V \tag{7}$$

$$\theta_m = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \tag{8}$$

and implemented in LabVIEW using the subvis shown in Table 3. The block diagram of the subvi that implements the plant model is shown in Figure 10.

The control structure that is used to illustrate the functionality of LabNS2 is shown in Figure 6. The LabVIEW subvi that implements the plant node is shown in Figure 11. The results are discussed in Section 8.

7.3. **Controller subvi.** The PID Controller used for the illustration of the emulator is designed for the plant described above using its transfer function model. The parameters for the Proportional, Integral and Derivative Gains are calculated using the Second Method of Ziegler Nichols.

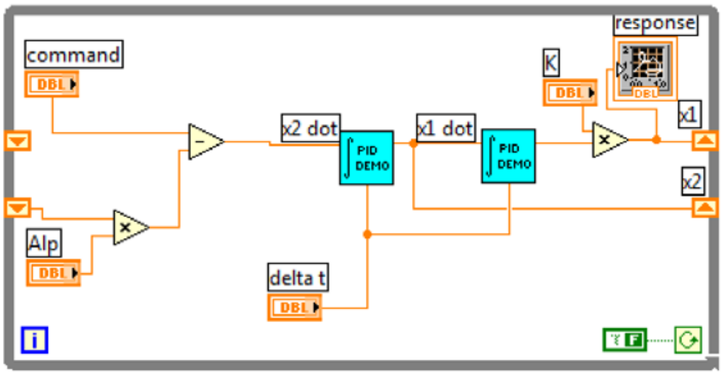


FIGURE 10. LabVIEW state space block diagram of the DC motor

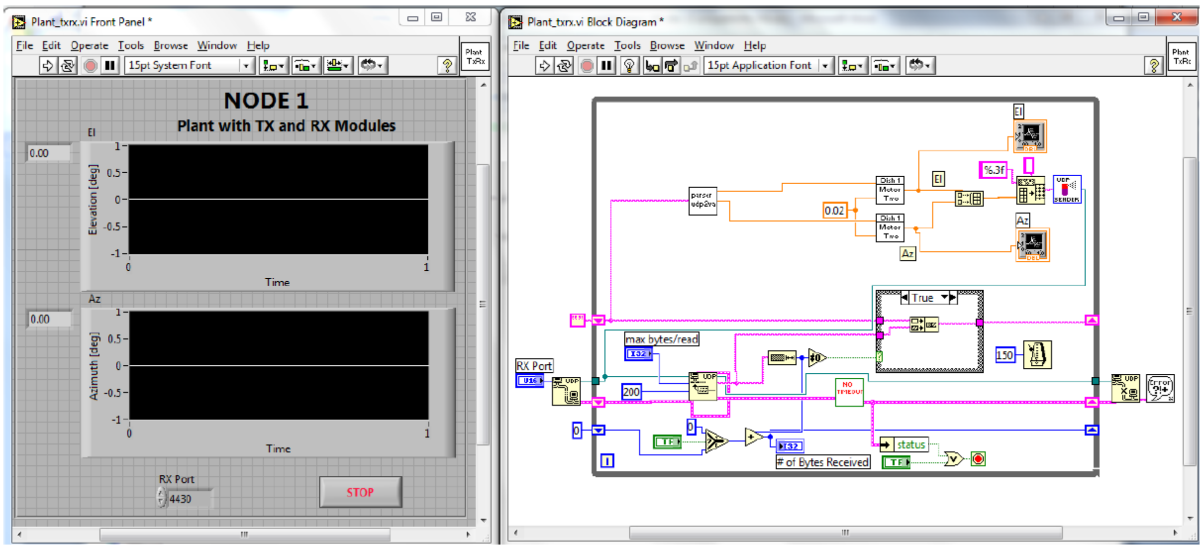

Plant_txrx.vi Front Panel *
File Edit Operate Tools Browse Window Help
15pt System Font
NODE 1
Plant with TX and RX Modules
El
Elevation [deg]
Time
Az
Azimuth [deg]
Time
RX Port
4430
STOP
Plant_txrx.vi Block Diagram *
File Edit Operate Tools Browse Window Help
15pt Application Font
True
max bytes/read
RX Port
200
150
status
# of Bytes Received

FIGURE 11. Plant vi with transmit and receive modules

The PID controller output is

$$u(t) = k_p e(t) + k_i \int_0^t e(t)dt + k_d \frac{de(t)}{dt} \tag{9}$$

where k_p is the Proportional gain, k_i is the integral gain, k_d is the Derivative gain and $e(t)$ is the error.

The transfer function of the PID controller is:

$$G_c(s) = \frac{k_d s^2 + k_p s + k_i}{s} \tag{10}$$

Equation (10) is implemented in LabVIEW using PID.vi subvi, and is shown in Figure 12.

The control structure that is used to illustrate the functionality of LabNS2 is shown in Figure 6. The LabVIEW subvi that implements the controller node is shown in Figure 13. The results are discussed in Section 8.

8. Operation Procedure and Results.

The high level flow chart of the operation of LabNS2 is shown in Figure 14. The flow chart is developed based on a typical networked control system shown in Figure 6.

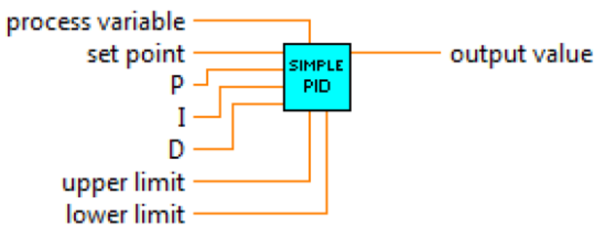


Figure 12. PID subvi

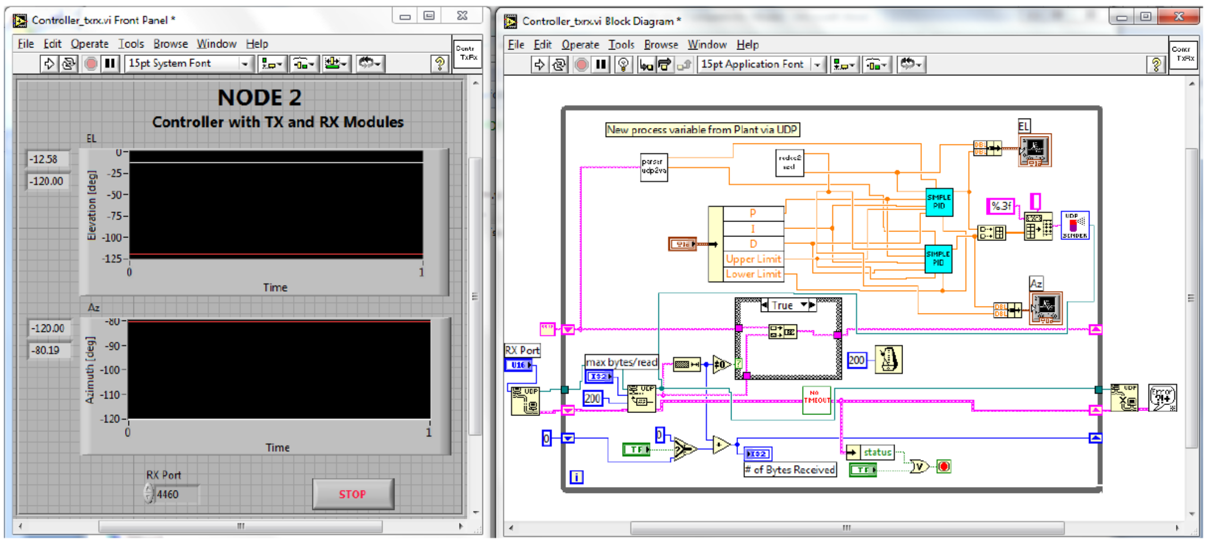


Figure 13. Controller vi with transmit and receive modules

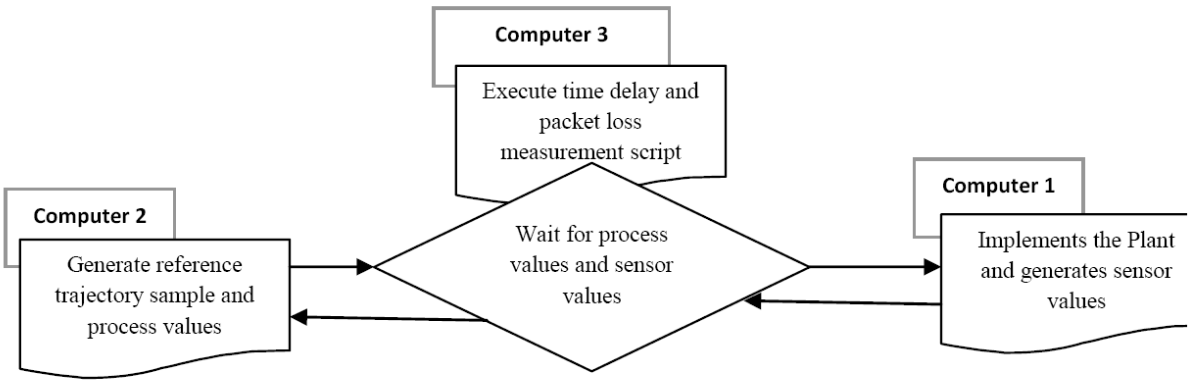

Computer 3
Execute time delay and packet loss measurement script
Computer 2
Generate reference trajectory sample and process values
Wait for process values and sensor values
Computer 1
Implements the Plant and generates sensor values

Figure 14. Flowchart of the operations

The reference computer generates the reference trajectory and connects to the controller via NS-2 time delay and packet loss measuring tool. The measured data is stored in the NS2 trace file. Using Cygwin, Tcl scripts are executed in the NS2 environment to generate NS2 trace files. The files are processed using the Tcl and Gawk scripts to calculate network induced time delays and packet loss.

To run LabNS2 start the following vi in LabVIEW^{TM}:

Dish_N_tf.vi (Antenna1.vi) – Plant and controller

Trajectory_rc_DM.vi

To run LabNS2 start the following scripts from the home directory in Cygwin:

cd tcl-scripts-cput/

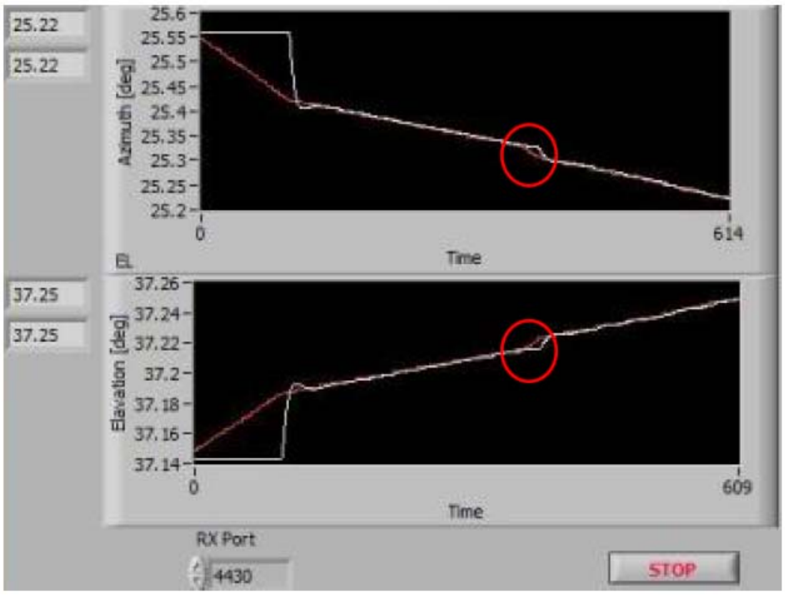
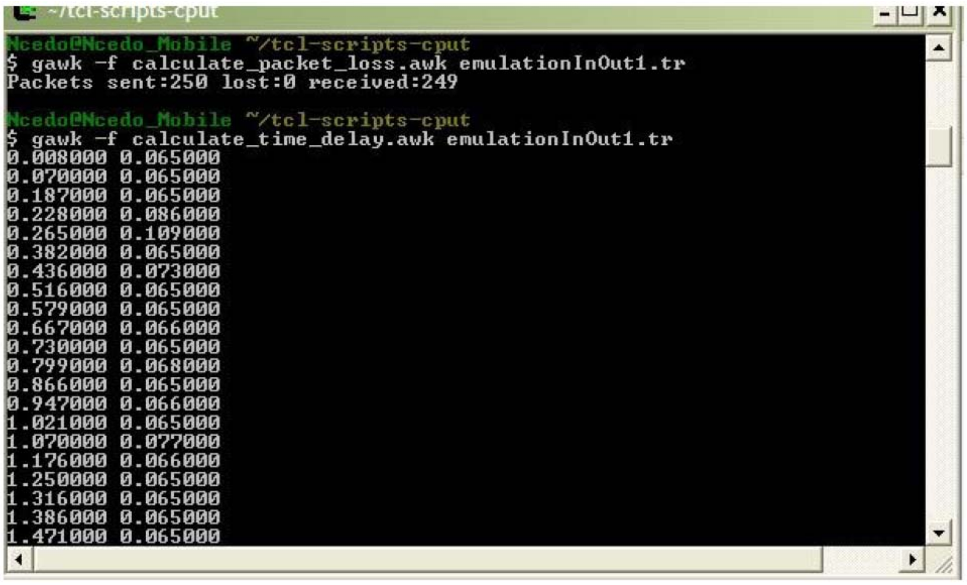


FIGURE 15. Snapshoot of the results



```
~/tcl-scripts-cput
Ncedo@Ncedo_Mobile ~/tcl-scripts-cput
$ gawk -f calculate_packet_loss.awk emulationInOut1.tr
Packets sent:250 lost:0 received:249

Ncedo@Ncedo_Mobile ~/tcl-scripts-cput
$ gawk -f calculate_time_delay.awk emulationInOut1.tr
0.008000 0.065000
0.070000 0.065000
0.187000 0.065000
0.228000 0.086000
0.265000 0.109000
0.382000 0.065000
0.436000 0.073000
0.516000 0.065000
0.579000 0.065000
0.667000 0.066000
0.730000 0.065000
0.799000 0.068000
0.866000 0.065000
0.947000 0.066000
1.021000 0.065000
1.070000 0.077000
1.176000 0.066000
1.250000 0.065000
1.316000 0.065000
1.386000 0.065000
1.471000 0.065000
```

FIGURE 16. LabNS2 emulation results calculated by the developed Gawk scripts

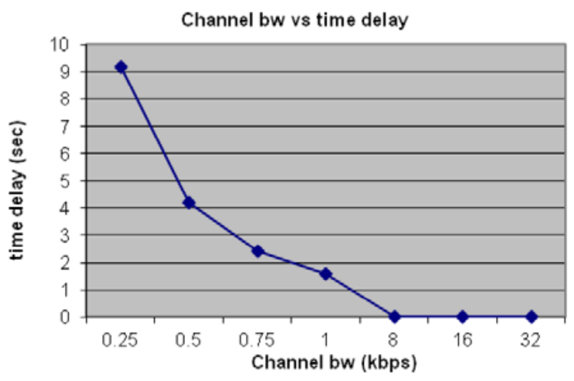
nse measure-tdpl-rc-a1.tcl <cbw> <ex_time> %Script to measure time delay and packet loss

To process the trace file, execute the Gawk script by using these commands:

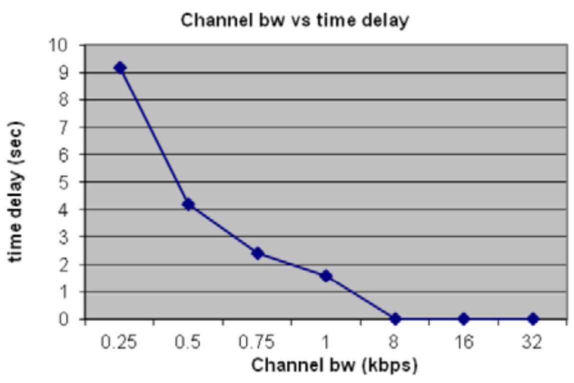
gawk –f <calculate_packet_loss.awk> <tracefile.tr>

gawk –f <calculate_time_delay.awk> <tracefile.tr > <excelfile.xls>

The snapshot of the results of the Plant model output connected through NS2 and via the communication network is shown in Figure 15. In the figure the two lines indicate the reference trajectory (white line) and the measurement output (red line). If one of the nodes is disconnected the error grows and the measurement output cannot track the reference trajectory as shown in Figure 15 (circled point). Note that as soon as the network is restored the measurement output tracks the reference trajectory.



(a)



(b)

FIGURE 17. LabNS2 emulation results. Processing results of the Gawk scripts. (b) is the zoomed portion of the channel bandwidth 8, 16 and 32kbps from (a).

The time delay and packet loss can be measured and calculated using LabNS2. The following experiment is performed to illustrate these measurements.

In the experiment LabNS2 is used to measure time delay between the controller and the actuator. The measured time delays based on the output trace file are shown in Figure 16. This figure shows how LabNS2 is used to investigate the relationship between the channel bandwidth and the network induced time delay in the NCS. The results show that time delay drops as the channel bandwidth increases, as shown in Figures 17(a) and 17(b).

The results of the packet loss counter are shown on the command prompt see Figure 16 row 3 from the top where 250 packets were sent, 0 packets were lost and 249 packets were received.

The values of the measured delays can further be used for calculation in real-time of the control action in some form of predicting time delays controllers.

9. **Conclusion.**

The paper presents a simple, scalable, and effective tool for simulation, emulation and real time implementation of NCS using LabVIEW^{TM}, and NS-2 (LabNS2). The architecture of LabNS2 is described and discussed. Tool Command Language (Tcl) and Gawk scripts are developed, presented and discussed. LabVIEW codes are developed, presented and discussed. Radio Telescope is presented as typical closed loop NCS. The implementation and operation procedure of the LabNS2 are described.

LabNS2 has four software modules that have to be executed to emulate the NCS environment. The modules are:

- Reference generating trajectory module implemented in LabVIEW. This module uses the developed software libraries described in Table 3 and it generates the desired positions for altitude and azimuth mounts of the dish antenna.
- Plant model and controller software implemented in LabVIEW. This module uses the developed software libraries described in Table 3. It receives desired reference positions via the network through NS-2 as shown in Figure 2. It realizes the controller and displays actual vs desired positions for both altitude and azimuth coordinates as shown in Figure 14.
- Network induced time delays and packet loss are measured using Tcl scripts. The scripts are listed in Table 2.

- Network induced time delays and packet loss are processed offline from the recorded output trace file using the developed script file in Gawk software. These scripts are listed in Table 2.

The complexity of measuring the communication network imperfection in a networked control system, including network induced time delays and packet loss makes it difficult for the control engineers to develop methods that can compensate or incorporate these imperfections in order to develop new control design methods for networked control systems. These network imperfections degrade the performance of the closed loop control system and result in a closed loop system instability. LabNS2 addresses this problem by providing an ideal environment that can be used to investigate the influence of network imperfections especially the network induced time delay and packet loss. It is done in a real-time environment by using tools that have been used for decades in their respective knowledge areas that are the information systems in the case of NS2 and the feedback control systems in the case of LabVIEW.

The LabNS2 toolkit has more capabilities than the existing till now tools for the NCS. For the case of the Modelica/NS2 the co-simulation is designed based on the simulation engine of NS-2 without using the emulation engine of NS-2 which uses the real-time scheduler to inject live data into the network [9].

In the paper both simulation and emulation engines are used to simulate and emulate the network using simulated data traffic and real-time data from the Ethernet via the sockets into the emulation environment of NS-2 thereby allowing for better understanding of influence of network induced time delays and packet loss effects on the NCS.

LabNS2 toolkit can be used successfully for simulation, emulation of hardware in the feedback control systems, and for real time system implementation. LabNS2 can also be used for educational purposes and research work in control systems, signal processing, communication systems, network systems, verification of mathematical models for communication networks (e.g., TCP models), and in the implementation of Lab experiments.

Acknowledgment. The work is funded by National Research Foundation under the THRIP grant No. TP2011061100004.

REFERENCES

[1] F. Wang and D. Liu, *Networked Control Systems Theory and Applications*, Springer, 2008.
[2] J. Y. Shin, J. W. Jang and J. M. Kim, The simulation and emulation verification that was based on NS-2, *International Journal of Computer Science and Network Security*, vol.9, no.3, pp.210-214, 2009.
[3] M. Pohjola and S. Nethi, *PiccSIM Manual*, Helsinki University of Technology, PiccSIM Toolchain Version 0.2, 2009.
[4] Y. Tipsuwan and M. Chow, Control methodologies in networked control systems, *Control Engineering Practice II*, pp.1099-1111, 2003.
[5] A. Friedl, S. Ubik, A. Kapravelos, M. Polychronakis and E. P. Markatos, Realistic passive packet loss measurement for high-speed network, *The 1st International Workshop on Traffic Monitoring and Analysis*, Aachen, Germany, 2009.
[6] N. S. Nise, *Control System Engineering*, 4th Edition, 2004.
[7] *Network Simulator NS-2*, www.ns.com.
[8] K. Fall and K. Varadhan, The ns manual, *The VINT Project*, 2008.
[9] A. Al-Hammouri, V. Liberatore, H. Al-Omari, Z. Al-Qudah, M. Branicky and D. Agrawal, A co-simulation platform for actuator networks, *Proc. of ACM SenSys*, Sydney, Australia, 2007.
[10] A. Cervin, Simulation of networked control systems, *The 3rd WIDE PhD School on Networked Control Systems*, Siena, Italy, 2009.
[11] *http://rtns.sssup.it/RTNSWebSite/RTNS.html*, 2013.
[12] G. Perbellini, *Network Advanced Modelling in NS-2*, Universita degli Studi di Verona, Facolta di Scienze MM.FF.NN, 2005.

[13] J. Yonan, The user-space VPN and open VPN, *Understanding the User-Space VPN-History, Conceptual Foundation, and Practical Usage*, 2003.

[14] *http://www.ephemeris.com/space-time.html*, 2013.

[15] *http://www.radiosky.com/radioeyesishere.html*, 2013.

[16] F. Lian, J. Moyne and D. M. Tilbury, Network control systems toolkit: A simulation package for analysis and design of control systems with network communication, *Technical Report UM-ME-01-04*, 2001.

[17] R. Yang, P. Shi and G. Liu, Filtering for discrete-time networked nonlinear systems with mixed random delays and packet dropouts, *IEEE Trans. on Automatic Control*, vol.56, no.11, pp.2655-2660, 2011.

[18] J. Nilsson, *Real-Time Control Systems with Delays*, Ph.D. Thesis, Lund Institute of Technology, 1998.

[19] M. Moayedi, Y. Foo and Y. Soh, Optimal and suboptimal minimum-variance filtering in networked systems with mixed uncertainties of random sensor delays, packet dropouts and missing measurements, *International Journal of Control, Automation, and Systems*, vol.8, no.2, pp.1179-1188, 2010.

[20] X. Li and S. Sun, Robust H_∞ control for networked systems with random packet dropouts and time delays, *International Workshop on Information and Electronic Engineering, Procedia Engineering*, vol.29, pp.4192-4197, 2012.

[21] T. Jia, Y. Niu and X. Wang, H control for networked systems with data packet dropout, *International Journal of Control, Automation, and Systems*, vol.8, no.2, pp.198-203, 2010.

[22] H. Yang, Y. Xia, P. Shi and M. Fu, Stability analysis for high frequency networked control systems, *IEEE Trans. on Automatic Control*, vol.57, no.10, pp.2694-2700, 2012.

[23] X. Luan, P. Shi and F. Liu, Stabilization of networked control systems with random delays, *IEEE Trans. on Industrial Electronics*, vol.58, no.9, pp.4323-4330, 2011.

[24] R. Yang, P. Shi, G. Liu and H. Gao, Network-based feedback control for systems with mixed delays based on quantization and dropout compensation, *Automatica*, vol.47, no.12, pp.2805-2809, 2011.

[25] H. Yang, Y. Xia and P. Shi, Stabilazation of networked control systems with nonuniform random sampling periods, *Int. J. Robust. Nonlinear Control*, pp.501-526, 2011.

[26] N. S. Mkondweni, *Design and Implementation of Linear Robust Networked Control Systems*, DTech Thesis, Cape Peninsula University of Technology, Bellville, Cape Town, 2013.