Applying Akaroa2 for Automated Simulation Length Control & On-Line Statistical Error Analysis of Results in OPNET

Mofassir Ul Haque
Computer Science and Software Engineering Dept
University of Canterbury
Christchurch, New Zealand
mofassir.haque@pg.canterbury.ac.nz

Krzysztof Pawlikowski
Computer Science and Software Engineering Dept
University of Canterbury
Christchurch, New Zealand
Krys.Pawlikowski@canterbury.ac.nz

Don McNickle
Computer Science and Software Engineering Dept
University of Canterbury
Christchurch, New Zealand
Don.McNickle@canterbury.ac.nz

Andreas Willig
Computer Science and Software Engineering Dept
University of Canterbury
Christchurch, New Zealand
Andreas.willig@canterbury.ac.nz

Abstract—**Simulation is used for developing and testing different scenarios under controlled and reproducible situations. Proper handling of the initial transient effect for steady state simulations, proper selection of simulation length and proper statistical analysis of results are essential for conducting credible simulations. Akaroa2, a universal controller for quantitative discrete event simulation, has been designed to improve credibility of simulation results. It automatically handles the initial transient effect, carries out statistical analysis of results and controls simulation length by stopping the simulation when the required precision is achieved based on continuous analysis of mean values. Simulation programs can be made to run under Akaroa2 to provide statistically valid results. OPNET is a popular simulator used for carrying out telecommunication and networking related simulations, but it does not provide automated simulation length control and initial transient effect handling. We have developed an interface using OPNET co-simulation capabilities to run OPNET simulations under Akaroa2 control. It will allow the user of OPNET to control the length of simulation on basis of the accuracy of statistical results. OPNET and Akaroa2 are available for free for non-profit research at universities.**

Keywords-Interface; simulation; Akaroa2; OPNET; Statistical Anylasis; co-simulation;

I. INTRODUCTION

Testbeds are useful tools for testing and validating new concepts. Cost of development/maintenance, lack of realism, inaccuracy and incompleteness of results are some of the problems of using testbeds [1]. Simulation offers a cheap alternative to constructing testbeds. It is a popular tool for validating networking and telecommunication concepts, and for conducting experiments that require repeatability and flexibility. OPNET, NS-2 and NS-3 are the most popular simulators being used for conducting simulation studies [2, 3]. None of these simulators provides support for automated simulation length control and initial transient effect handling. One needs to repeat a simulation experiment a number of times with different seeds of the random number generator and then calculate the confidence interval (CI) for a prescribed confidence level (CL). The final results produced by these simulators are not valid without calculation of statistical errors.

One of the problems identified with regard to credibility of simulation studies of telecommunication network is related to proper analysis of simulation results in cases where output data are correlated. The inference drawn from simulation results could be misleading if they are not analyzed properly. A survey of 2246 papers published in top telecommunication related conferences revealed that 76.45% of papers reported purely random results or their final results were without proper statistical analysis [4].

Proper selection of the simulation experiment length is also extremely important. Simulation experiments run for a short period of time can give misleading results whereas simulation experiments run for a longer period of time will result in wastage of computing resources. It is impossible to assess statistical errors in advance, for pre-determined lengths of simulation experiments. Study of network behavior in a steady state is even more difficult as simulation takes some time to reach a steady state. The data collected during an initial transient time can add bias to the final results. It is therefore important that data collected during the initial transient is accurately identified and properly handled [5].

The Akaroa2 software developed by the Simulation Research Group at University of Canterbury, New Zealand solves these problems by automating the simulation length control and initial transient effect handling process [6]. It decides about the duration of simulation experiment by stopping it when the results become sufficiently accurate, at the confidence level specified by user. In case of steady-state

978-1-4673-4410-4/12/$31.00 ©2012 IEEE

analysis, it discards samples collected during the initial transient period, before initiating on-line analysis of output data in steady state. It supports terminating and steady-state simulations. In the former case, the procedures for sequential mean and variance analysis are described in [7, 8-9], while the procedure adopted for terminating simulations is presented in [10].

In this paper, we report on controlling OPNET simulations by Akaroa2, by using a specially designed Akaora2-OPNET interface. By running OPNET simulations under Akaroa2 control, we are able to control the simulation length and generate credible final results, following fully automated on-line error analysis.

The rest of the paper is organized as follows. Section II of the paper defines Akaroa-2. Sections III explicate co-simulation capability of OPNET. Overview of the interface between OPNET and Akaroa-2 is given in Section IV. Section V provides results of running MM1 simulation in OPNET under Akaroa-2 control before the paper is finally concluded in Section VI.

II. AKAROA-2

Akaroa2 is a universal software controller for simulation, able to perform on-line analysis of statistical errors of simulation results and to speed up simulation using the Multiple Replication in Parallel (MRIP) technique. Akaroa2 stops the simulation when the required precision of results is achieved, by conducting sequential estimation of mean values and variances, and their statistical errors. The user can specify the required confidence level and statistical precision for each observed parameter. Akaroa2 is written in the C++ language. It can run on a single or multiple computers distributed across the globe or in a local area network. Running Akaroa2 on PlanetLab nodes, distributed across the globe, is also possible [11]. Akaroa2 is widely used by the simulation community around the globe; its records of the last 12 years show over 4600 downloads of the software by users from over 80 countries (as shown on <www.akaroa.canterbury.ac.nz>, in July 2012). The current version allows automated analysis of mean values, proportions and probabilities.

Akaroa2 can be used in stand-alone or Multiple Replication in Parallel (MRIP) mode. In MRIP mode, multiple instances of the same simulation are run in parallel on different machines until a specified level of accuracy of the results is met. In standalone mode, the Akaroa2 analysis engine can be invoked directly by passing the parameter to be observed using the AkObservation (value) procedure. Figure 1 shows the final output produced by the Akaroa2 analysis engine after achieving required precision.

Figure 1. Akaroa2 Output

The explanation of different terms used in Akaroa2 output is given in Table I.

TABLE I. AKAROA2 OUTPUT VARIABLES

Parameter	Explanation
Param	Number of parameters observed
Estimate	Final estimate of analysed parameter
Delta	Half width of confidence interval (Margin of Error)
Conf	Desired confidence interval
Var	Variance of estimate
Count	Total number of observation analyzed
Trans	Total observations discarded during the transient phase

Other network simulators, such as NS-2, NS-3, etc. can be made to work under Akaroa2 for providing reliable results too. A simulation program written by means of a simulator has to be recompiled with Akoara2 libraries. The stream of observations produced by the simulation program will be passed to Akaroa2 using the AkObservation (value) procedure call. AkObservation continuously checks if the required precision for the estimate has been achieved. Akaroa2 will produce the final results and stop execution of a given simulation once the required accuracy is achieved. More than one parameter can be analyzed by declaring their number using AkDeclareParameters(n) and differentiating their observations by means of AkObservation(i, value) function.

III. OPNET

OPNET Modeler provides excellent support for modeling communication networks e.g. WLAN, UMTS, LTE, Wi-Max, MANET etc. It is a discrete-event simulator which is used for simulation studies aimed at performance analysis, capacity planning and design of new protocols [12]. It uses a hierarchal modeling structure which consists of network, node and process domains. The network domain is a high level description of objects. The node domain specifies the internal structure of network nodes. The process domain allows us to define behavior of process modules which exist in the node domain, using state transition diagrams and C code.

OPNET Modeler provides a co-simulation capability for interacting with external hardware or software systems [13]. An OPNET simulation model is linked with external modules using the External System Interface. Co-simulation can either allow OPNET or the external program to control the execution of a simulation. The co-simulation process requires use of the following five components:

- Simulation description file
- External System Definitions (ESD) model
- External System interface (Esys)
- External Simulation Access (ESA) API
- Esys API

The simulation description file is primarily used to define how to build a simulation, and it includes the location of libraries required for linking an external program with OPNET. It is a text file with .sd extension. The External System Definitions (ESD) model is used for defining interfaces which are used for reading and writing values between OPNET and an external program. The External System Interface (Esys) is a

node domain module with a process model for communicating with an external system. The External Simulation Access (ESA) API provides functions for reading and writing values on interfaces between OPNET and an external program from external code. The Esys API provides functions for accessing interfaces between OPNET and an external program from an OPNET process model. The Figure 2 shows co-simulation at a conceptual level, if OPNET cooperates with Akaroa2

Figure 2. Co-Simulation of OPNET and Akaroa2

IV. CO SIMULATION INTERFACE

Queues are the integral part of any data network as packets arrive at different points, wait in various queues, and get service and exit. Queuing theory is widely used in computer and network performance analysis [14]. A simple M/M/1 queue model shown in Figure 3 was created for simulation in OPNET Modeler. It was ensured that the queue reaches steady-state for the assumed distribution of packet sizes, as well as arrival and service rate. The main purpose was to create and run a simulation in OPNET and to use Akaroa2 for stopping the simulation when results become sufficiently accurate.

A. M/M/1 Queue Node Model

The M/M/1 queue model consists of a traffic generator, a queue and an Esys module. The packets' inter arrival times are exponentially distributed. The queue is equipped with a very large (infinite) buffer and a processor. The Esys module destroys a packet after its service, to free up memory, calculates the packet's delay and passes this value on to Akaroa2 for processing. Packet streams are used to connect these three components.

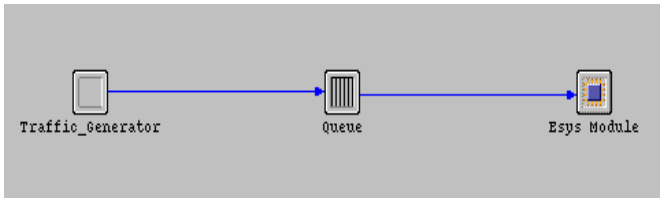

Figure 3. M/M/1 Queue Node Model

B. Simulation Description File

The simulation description file contains information for building and linking an executable file. We ran the co-simulation under the 64-bit Linux operating system. Either OPNET or external program i.e. Akaroa2 can control the flow of simulation. If use_esa_main is set to No then external program drives the co-simulation flow. The simulation was being controlled by Akaroa2. Therefore, we set use_esa_main to no. The locations of libraries required for running Akaroa2 are defined in this file. The contents of the simulation description file is shown in Figure 4.

```
# Simulation Description
start_definition
        platform:       linux_x86
        use_esa_main:   no
        bind_obj:       cosim_17_ak_mm1_external_code1.dev64.i1.ex.o
        #kernel:        develpoment
        bind_lib:       /usr/local/akaroa/lib/libargs.a -lfl
        bind_lib:       /usr/local/akaroa/lib/libakaroa.a
        bind_lib:       -lfl
        #bitness:       64bit
        #dll_lib:       libakaroa.so
        #dll_lib:       libopsim.so
end_definition
```

Figure 4. Simulation Description File

C. External System Definition

The External System Definitions (ESD) model is shown in Figure 5. The upper part contains the name of the simulation description file to be used with this model. The lower part defines the Esys interface name, type and direction for data transfer. We have defined two uni-directional variables called delay and stop. The variable delay is used to pass a delay value calculated by Esys in its process model to Akaroa2 for processing and the stop variable carries an interrupt signal from Akaroa2 to OPNET.

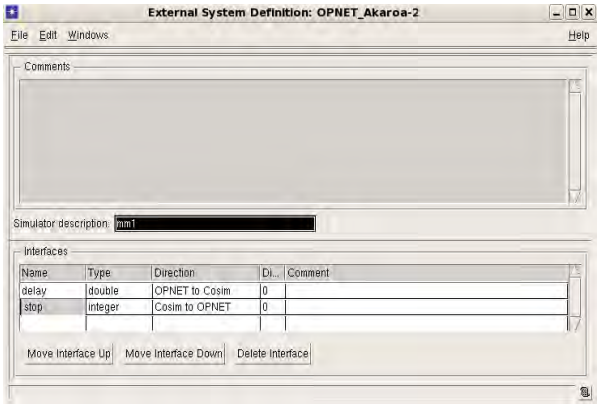

Figure 5. ESD Model

D. Process Model for Esys

The process model for Esys is shown in Figure 6. It calculates the queuing delay by using equation 1. Here, the op_sim_time () function is used to get the current simulation time, and the op_pk_create_time_get() function is used to get the time when this packet was created. After calculating the delay, it uses the Esys API function op_esys_interface_value_set() for immediately writing these values to interface.

delay = op_sim_time() – op_pk_create_time_get(); (1)

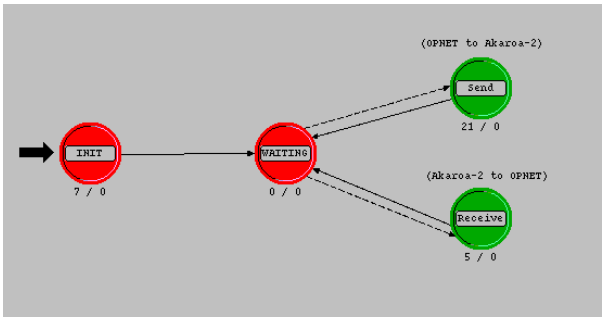

Figure 6. Process Model for Esys

E. Control Program

The control program is written in C. It uses the system API to handle the received delay value from OPNET. The function Esa_Interface_Value_Get() is used to read values from the interface and make a call back to a callback() function. The call back function passes the read values to Akaroa2 engine for evaluation using AkObservation(value) function. When the desired accuracy is achieved then Akaroa2 stops the OPNET simulation and produces the final output results.

F. Simulation Termination

When Akaroa2 has collected enough observations it terminates the OPNET simulation by sending the SIGTERM (Terminal) signal to it. OPNET expects the SIGINT (Interrupt) signal for graceful closure and proper storage of simulation results. Therefore, the Akaroa2 SIGTERM signal has to be replaced by the SIGINT signal. The C language provides support for catching and handling signals. The signal handling function has to be registered with the kernel. In our control program, we have used the on_exit() and raise() functions of C to catch Akaroa2 SIGTERM signal and to generate a SIGINT signal for graceful closure of OPNET simulation.

G. Running the Co-Simulation

External code, OPNET simulation code, associated models and the required libraries have to be built into one executable file. The OPNET Op_mksim command is used to bind all elements into one executable file. This executable file can then be run from command line or from OPNET graphical user interface.

V. Simulation Results

To demonstrate the advantages of using Akaora2 with OPNET, we have executed simple simulations of an M/M/1 system. The M/M/1 system consists of a packet generator and a server with unlimited buffer capacity, as depicted in Figure 3. We estimated the steady-state mean response (mean time spent by a packet in the queue and in service) assuming that arriving packets form a Poisson process with $\lambda = 1$ packets per second. The calculated value for mean packet delay comes to 15 second. The packet sizes followed an exponential distribution with mean outcome set to 9,000 bits. The service rate was set to 9,600 bits per second. This means that the M/M/1 queuing system was loaded to 93.75%.

The simulation process was stopped when the estimate of the steady-state mean response reached a relative statistical error not greater than 5%, at a confidence level of 0.95. The M/M/1 simulations in OPNET were run under the control of Akaroa2, i.e. simulations were stopped by Akaroa2 when enough observations were collected. We have used OPNET Modeler version 17.1 and Akaroa2 version 2.7.10 for our simulation experimentation. Both programs were installed on Red Hat Linux version 2.16.0. OPNET Modeler version 17.1 and Akaroa2 version 2.7.10 are available for free use for non-commercial research at universities.

The M/M/1 simulations were initially run for short (simulated) times without Akaroa-2's run time control. The results are shown in Table II. The final estimated values of the mean time spent by a packet in queue are much lower than the actual value. This is due to relatively small values observed during the transient period, as shown in Figure 7. These results show that, without proper run control and proper handling of the initial transient effect, one is likely to get incorrect and misleading results.

TABLE II. Incorrect Results Due To Short Run

Time in Hours	Mean Packet Delay (Seconds)
15 Minutes	8.511
30 Minutes	9.252
1 Hour	9.271
5 Hours	9.814
20 Hours	10.82

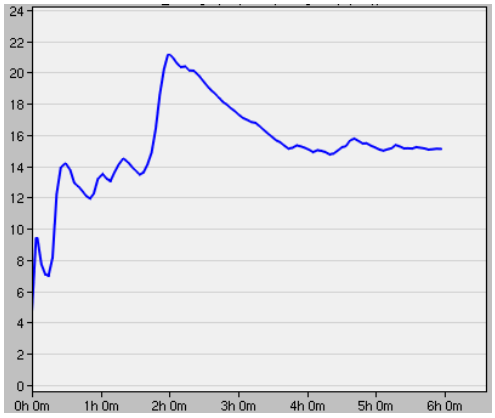

Figure 7. The running mean of response time

The mean time spent by a packet in queue from M/M/1 simulations, run under control of Akaroa2 is given in Table III. It shows mean packet's delays in steady-state, the half-width of its confidence interval (Delta), variance of the estimate, the sample length (count) and the number of observations that were deleted since they were collected during the initial transient phase (Trans).

TABLE III. Result For M/M/1 Queue Obtained by Akaroa2

Seed	Akaroa2/OPNET Results				
	Estimate	Delta	Variance	Count	Trans
193163244	15.434	0.91432	0.149431	2,86,082	442
379814045	14.5107	0.805152	0.115878	2,21,563	507
434868289	15.8224	0.938817	0.157545	4,35,015	327
787468622	15.7622	0.930089	0.15463	2,88,927	1189
933588178	15.5136	0.893512	0.142707	1,60,056	351

One can see that the estimate obtained by running OPNET under Akaroa2 is more conservative and accurate as their statistical errors have been controlled. The estimates obtained by running OPNET under Akaroa2 lie within the confidence interval bracket of calculated value.

VI. Conclusions

It is important that simulation results are analyzed using proper statistical techniques. Confidence intervals are commonly used to express the degree of accuracy of the final results. These result values must lie between upper and lower bound of the confidence interval. For steady state simulations,

the length of the initial transient needs to be correctly identified and all the observations collected during that time should be discarded. Akaroa2 decides about the duration of a simulation experiment by on-line analysis of confidence intervals at confidence levels specified by users and discards observations collected during the initial transient period. It reports the final values of performance measures under investigation along with their variances, and the margins of error. In this paper, we have developed an interface to run simulations created in OPNET under Akaroa2.

Another important feature supported by Akaroa2 is MRIP (Multiple Replications in Parallel). In MRIP, multiple processors run their own replications of sequential simulation, but cooperate with central analyzers (one central analyzer for each performance measure analyzed) that are responsible for analyzing the results and stopping the simulations when the specified level of accuracy is met. The MRIP technique can significantly speed up simulation if replications are launched on a larger homogeneous set of computers. In future, we plan to extend our co-simulation framework to enable MRIP support for OPNET simulations.

ACKNOWLEDGMENT

Our thanks to OPNET Technologies, Inc for supporting our research by providing OPNET modeler under university research program.

REFERENCES

[1] D.R.Chones and F. E. Bustamante, "Pitfalls for Testbed Evaluations of Internet Systems," SIGCOMM Comput. Commun. Rev., vol. 40, pp. 43-50, Apr. 2010.

[2] Jianli Pan, Raj Jain,"A Survey of Network Simulation Tools: Current Status and Future Development", Tech Rep,Washington University in St. Louis,Computer Science & Engineering Department, 2008.

[3] G. F. Lucio, M. Paredes-Farrera, E. Jammeh, M. Fleury, and M. J.Reed, "OPNET modeler and Ns-2: Comparing the Accuracy of Network Simulators for Packet-Level Analysis Using a Network Testbed," in In 3rd WEAS International Conference on Simulation, Modelling and Optimization (ICOSMO), pp. 700-707, 2003.

[4] K. Pawlikowski, H. D. J. Jeong, and J. S. R. Lee, "On Credibility of Simulation Studies of Telecommunication Networks," IEEE Communications Magazine, vol. 40, no. 1, pp. 132-139, 2002.

[5] White, Jr, K.P. and Robinson, S.,"Initial Transient Period in Steady-State Systems" Encyclopaedia of Operations Research and Management Science. Wiley Encyclopedia of Operations Research and Management Science(J.J.Cochran,ed.),Wiley,NewYork.DOI:10.1002 /978047040053. eorms0408, Feb 2011.

[6] G.Ewing,"ProjectAkaroa"."http://www.akaroa.canterbury.ac.nz" Accessed on: July 19, 2012:

[7] K. Pawlikowski, V. W. C. Yau, and D. McNickle, "Distributed Stochastic Discrete-Event Simulation in Parallel Time Streams," in Proceedings of the 26th Conference on Winter simulation, WSC '94, (Orlando, Florida, United States), pp. 723-730, Society for Computer Simulation International, Dec 1994.

[8] Ewing,G.,Pawlikowski,K.: Spectral Analysis for Confidence Interval Estimation Under Multiple Replications in Parallel. In: 14th European SimulationSymposium. pp.52-61,ISCS Press,Dresden, Germany, 2002.

[9] N. Shaw, McNickle, D., Pawlikowski, K.: Fast Automated Estimation of Varience in Sequential Discrete Event Stochistic Simulation. In: 25th European Conference on Modelling and Simulation, Krakow, Poland, 2011.

[10] Pawlikowski, K., Ewing, G., McNickle, D.: Performance Evaluation of Industrial Processes in Computer Network Environments. In: European Conference on Concurrent Engineering, pp. 129-135. Int. Society for Computer Simulation, Erlangen, Germany (1998)

[11] Mofassir Haque, Krzysztof Pawlikowski, Don McNickle, and Gregory Ewing. World-Wide Distributed Multiple Replications in Parallel for Quantitative Sequential Simulation. In Proceedings of the 11th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part II, ICA3PP'11, pages 33–42, Melbourne, Australia, Oct 2011. Springer-Verlag.

[12] P. Oenek, " Anyalsis of the Network Communication With OPNET," in Proceedings of the 16th International Conference on Systems Science: Volume II, pp. 312-317, 2007.

[13] M. Bartl, J. Hosek, T. Matocha, K. Molnar, and L. Rucka, " Integration of real network components into OPNET modeler co-simulation process," WTOC, vol. 9, pp. 553-562, Sept. 2010

[14] Julio Rojas-Mora, Eitan Altman, and Tania Jimenez, "Some considerations in simulating an M/M/1 queue," In Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS '09). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, Belgium, 2009.