Networked Control System Wind Tunnel (NCSWT)- An evaluation tool for networked multi-agent systems

Derek Riley
Middle Tennessee State University
Murfreesboro, TN
ddriley@mtsu.edu

Emeka Eyisi
Vanderbilt University
Institute for Software and Integrated Systems
Nashville, TN
emeka.p.eyisi@vanderbilt.edu

Jia Bai
Vanderbilt University
Institute for Software and Integrated Systems
Nashville, TN
jia.bai@vanderbilt.edu

Xenofon Koutsoukos
Vanderbilt University
Institute for Software and Integrated Systems
Nashville, TN
xenofon.koutsoukos@vanderbilt.edu

Yuan Xue
Vanderbilt University
Institute for Software and Integrated Systems
Nashville, TN
yuan.xue@vanderbilt.edu

Janos Sztipanovits
Vanderbilt University
Institute for Software and Integrated Systems
Nashville, TN
janos.sztipanovits@vanderbilt.edu

ABSTRACT

Cyber-physical systems, such groups of unmanned aerial vehicles, are often monitored and controlled by networked control systems (NCS). NCS are deployed in many environments subject to realistic, complex network interactions, so evaluation of NCS is crucial to ensuring that NCS function as intended. Given the varied nature of NCS, it is appropriate to use a heterogenous simulation environment to capture the dynamics; however, the design and integration of heterogeneous simulation environments is a complex problem. In this work we present the Networked Control System Wind Tunnel (NCSWT), an integrated simulation environment for NCS. The NCSWT integrates MATLAB/Simulink and ns-2 according to the High Level Architecture standard. We demonstrate the convenience and efficiency of the NCSWT using several case studies where realistic network effects such as data drops and delays are introduced. We also demonstrate the flexibility and power of the tool in modeling realistic NCS.

Keywords

Modeling, Simlation, Networked Control Systems

1. INTRODUCTION

The integration of physical systems through computing and networking has become one of the most pervasive applications of Cyber-Physical Systems (CPS). This paper is inspired by the rapidly increasing use of Networked Control System (NCS) architectures in constructing real-world CPSs that integrate computational and physical devices using wireless networks such as medical device networks or groups of unmanned vehicles. NCS research is an active area, investigating problems at the intersection of control systems, networking, and computer science [2].

In a conventional design flow for NCS, controller dynamics are often synthesized with the purpose of ensuring stability and optimizing performance. Typically, control design is performed without considering implementation side effects (e.g. time varying delays caused by network effects, timing accuracy caused by shared resource and schedulers, etc.); however, these effects can be significant on the stability and performance of the system. Therefore, it is important to develop accurate NCS modeling and simulation tools to evaluate the effect caused by these phenomena.

Several powerful simulation tools for modeling control systems and networks exist such as Matlab/Simulink [15], a software tool extensively used for control design and simulation of control systems, and ns-2 [19], the most widely used open-source network simulator. Individually, these tools can model NCS systems, but when combined, they can leverage the benefits of the individual simulation engines to model larger, more realistic systems. However, significant challenges such as the reconciliation of time synchronization and data communication between the simulation tools need to be addressed.

Our integration tool, called the NCS Wind Tunnel (NCSWT), provides state-of-the-art capabilities for combining control and network abstractions for NCS rapid prototyping, design, and simulation. The NCSWT provides a formal gateway for coordination and synchronization between Matlab/Simulink and ns-2. To enable time synchronization between Matlab/Simulink and ns-2 network simulation engine, the time management mechanisms of both simulators are modified to conform to the High Level Architecture (HLA) standard [13]. The HLA is a framework that provides support for the integration at the API level and interaction levels useful for coordinating data communication and time synchronization.

Another contribution of this work is a demonstration of the NCSWT tool using several case studies. We consider three NCS scenarios. The first scenario is a simple NCS

SIMUTOOLS 2011, March 21-25, Barcelona, Spain
Copyright © 2011 ICST 978-1-936968-00-8
DOI 10.4108/icst.simutools.2011.245514

composed of a plant and controller communicating through a UDP connection over an IEEE 802.11-based wireless network. This scenario is used to demonstrate the accuracy and convenience of using the NCSWT. The second scenario is a remote-controlled unmanned aerial vehicle (UAV), which is controlled over a wireless network to follow a given reference trajectory. This scenario is used to evaluate network phenomena such as packet loss, delay and multi-hop topologies for a safety critical system. The third scenario is a network of three plant-controller pairs sharing a communication channel and operating at different sampling rates. This scenario is used to demonstrate the flexibility and scalability of the NCSWT in handling realistic, heterogeneous models.

This paper is organized as follows. The second section presents the related work, the third section presents the NCSWT simulation framework, the fourth section presents the case studies, the fifth section presents an evaluation of the tool, and the final section concludes the work.

2. RELATED WORK

Different approaches are often sought in the simulation of NCS. Some general network simulators such as ns-2 and OMNeT++ [19, 4], which are discrete-event simulators, are often used. These simulators generally lack the ability to simulate continuous-time dynamics as well as the flexibility and ease of modeling complex dynamical systems. The simulators presented in [6, 1, 9] are NCS simulators that do not deliver any control design support and cannot accurately model complex dynamical systems.

Rather than model NCS using general network simulators, some approaches co-simulate NCS using powerful simulators for dynamical systems and then incorporate network abstractions. A typical example of this approach is Truetime. Truetime extends Matlab/Simulink with platform-related modeling concepts (i.e., networks, clocks, schedulers) and supports simulation of networked and embedded control systems with implementation effects [3]. Modeling the network dynamics in Truetime is highly abstracted, and therefore, limits the level of details of the network layers that can be modeled and implemented. For example, in TrueTime it is not straightforward to specify routing protocols or network queuing management schemes.

Recently there have been advances in various approaches that integrate or couple multiple simulators together in order to effectively simulate NCS. In [11], a tool chain called PiccSIM was developed, which allows the integration of Matlab/Simulink models with ns-2. PiccSIM provides a graphical user interface for the design of networked control systems and the automatic code generation of ns-2 scripts and Matlab/Simulink models. Also in [7], a special simulator coupling concept implemented in C/C++ is used to integrate the simulators ModelSim, Matlab/Simulink and ns-2. Although these two approaches address integration of simulators, the integration is not based on a standard framework, so it is not possible to reason about the accuracy or extensibility of the approaches.

The HLA is a standard for simulation interoperability originally developed by the U.S. Department of Defense [8]. IEEE now oversees the ongoing evolution of the architecture as the HLA Standard 1516. The HLA consist of three components: (1) Federation Rules that define the basic principles underlying the HLA and describe the simulation and federate responsibilities; (2) Interface Specification that defines the interface to the Run-Time Infrastructure (RTI) services that provide the means for simulators to coordinate the execution and exchange of information; (3) Object Model Template (OMT) that provides a standard format for describing information of common interest to the federates. An HLA simulation typically consists of a collection of simulators - each simulator is called a federate. Communications between different federates is managed by the Run-Time Infrastructure (RTI). The RTI provides a set of services such as time management, data distribution, and ownership management. Since HLA is an accepted standard, a number of RTI implementations are available [13].

An HLA simulation framework as well as code generation tools were developed for the Command and Control Wind Tunnel (C2WT) project, which is a robust multi-model simulation framework for integrating heterogeneous simulation components using the RTI within the HLA platform [17, 18]. The C2WT captures not only the necessary interface specification for running heterogeneous simulations over HLA, but also a variety of mechanisms for configuring, enhancing, and detailing the simulation execution. The C2WT framework uses the discrete event model of computation as the common semantic framework for the precise integration of an extensible range of simulation engines. To integrate the models, the C2WT uses metamodeling [5] and the metaprogrammable MIC tool suite [10] for developing a Model Integration Layer [16], which can be used to formally define the simulation semantics. The NCSWT extends the capability of the C2WT by providing an interface for ns-2 to interact with the rest of the C2WT system. This allows existing C2WT scenarios to leverage the accuracy of ns-2 for network simulation.

3. NCSWT INFRASTRUCTURE

Modeling of NCS requires combining heterogeneous simulation tools, each of which potentially could have its own modeling formalism or underlying model of computation. In this section we present the Network Control System Wind Tunnel (NCSWT), which integrates the ns-2 simulator with MATLAB/Simulink models into the C2WT infrastructure [17]. The NCSWT enforces the HLA time synchronization and data sharing standards to ensure accurate time synchronization and data passing between the heterogeneous models. This integration allows for the modeling of NCS in the presence of realistic network phenomena by ns-2 such as variable packet delays and losses. The NCSWT addresses complexities inherent in integrating simulators by use of well-defined models and by providing a model integration layer that enables rapid integration and configuration of simulators.

The NCSWT framework uses the discrete event model of computation as the common semantic framework for the precise integration of simulation engines. The simulators, including Matlab/Simulink and ns-2, are integrated with the Run-Time Infrastructure (RTI) according to the HLA standard. Each simulation model, when incorporated into the overall simulation environment, requires integration on two levels: the interaction level, and the API level. Interaction level integration addresses the issues of time synchronization and coordination, whereas API-level integration provides data passing services. The NCSWT offers a solution for multi-model simulation by decomposing the problem into model integration and experiment integration tasks.

3.1 Model Integration

Figure 1 shows the simulation architecture of the NC-SWT. The metamodeling and the metaprogrammable tool suite of the C2WT are used for developing the model integration layer. In the model integration layer, a designer can formally specify the model integration using a carefully selected collection of modeling concepts in the Model Integration Language [17]. These concepts help to specify the overall scenario as well as the individual simulation components and how they communicate. In addition, integration can be designed by describing the data representation and data flow elements in the tool suite. In Figure 1, the control system models, designed in Matlab/Simulink, define the dynamical systems and digital controllers to be simulated. The network models, designed in ns-2, define the network topology as well the properties of the network used for the communication of the control system. The Simulink and NCSWT federates essentially represent the communication interfaces for the control system models and network models respectively using the HLA simulation integration platform.

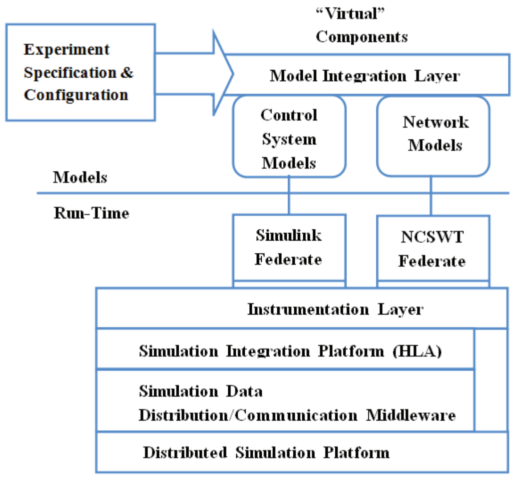

"Virtual" Components
Experiment Specification & Configuration
Model Integration Layer
Control System Models
Network Models
Models
Run-Time
Simulink Federate
NCSWT Federate
Instrumentation Layer
Simulation Integration Platform (HLA)
Simulation Data Distribution/Communication Middleware
Distributed Simulation Platform

Figure 1: NCSWT Simulation Architecture

3.2 Experiment Integration

The NCSWT uses the model interpretation infrastructure provided in the C2WT tool suite to generate some of the code necessary for integrating the heterogeneous experiments. After specifying the details of each experiment, the model interpreter generates several configuration files. These files configure the HLA API, configure simulators like Matlab, and also create Java and C++ skeleton code for run-time use.

Control system models, including the plant and controller subsystems, are designed in Simulink, and network models are modeled in ns-2. Existing C2WT tools [17] provide some of the framework for the Simulink federate integration using the HLA standard, but modifications are made to the existing integration tools for time synchronization to improve precision and allow for finer-grained time steps. In this work, we create a new NCSWT federate to integrate the ns-2 network models.

The NCSWT federate interface for is written for Windows (using C++) and utilizes associated libraries and classes from the C2WT [17] to synchronize with the RTI using the HLA standard. It acts as a portal to the C2WT system for ns-2 using sockets by enforcing the time synchronization and data passing standards to allow ns-2 to communicate with the RTI.

Figure 2 shows the communication overview of the NC-SWT federate interface for ns-2. PC1 (Windows) is used to execute the RTI as well as the NCSWT federate interface while PC2 (Linux) executes ns-2. The NCSWT federate interface uses sockets to communicate with ns-2. The socket connections on PC2 are established with synchronous sockets to ensure that the time synchronization and data passing are synchronized between the RTI and ns-2 according to the HLA standards.

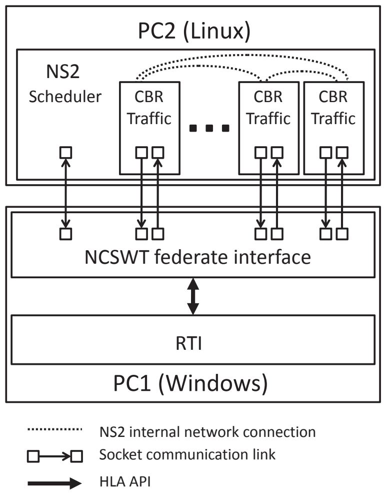


Figure 2: NCSWT communication overview

3.3 Time Synchronization

Time synchronization between various simulation actors is critical to preserve causality with simulations potentially operating in different time scales or time models. The NC-SWT implements the HLA time synchronization protocol to precisely maintain the same logical time as ns-2.

Because the ns-2 simulator uses a discrete event model of computation, time synchronization can be incorporated along with event scheduling. The scheduler class in ns-2 is modified to block scheduling of new events until it receives an adequate time advance grant (TAG) from the RTI. The ns-2 scheduler submits a time advance request (TAR) once it is ready to execute an event scheduled for at a time later than the latest TAG. The NCSWT passes the TAG and TAR messages between ns-2 and the RTI via socket connections.

An example time synchronization scenario is depicted in Figure 3. An initialization TAR is passed from ns-2 through the NCSWT to the RTI, and once a TAG is received, the ns-2 scheduler can unblock and run until it reaches a synchronization event (an operation requiring synchronization with the RTI). At that point, a TAR is submitted through the NCSWT to the RTI and ns-2 blocks. It is possible that the newest TAG will be less than the most recent TAR (i.e. $t_2 < t_1$), so ns-2 only executes events scheduled for times less than or equal to the latest TAG.

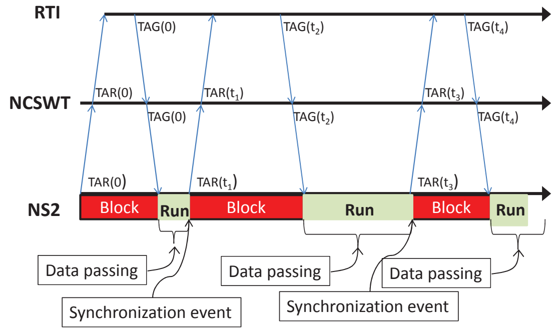


Figure 3: Time synchronization between the ns-2 and the RTI

3.4 Data Communication

The ability for data to be passed precisely between NCS components is crucial to an accurate simulation. The HLA standard specifies an API for data sharing including subscribing and publishing data [8], and the NCSWT conforms to the standard. First, we will describe how networks are modeled in our tool, and then we will describe how data is passed.

The network connections between components must be modeled both by the NCSWT and the ns-2 models to ensure the model is consistent. Modeled network connections are created in the NCSWT federate model, but no details about the type (wireless or wired), speed, or other attributes are modeled in the NCSWT federate. The details of the network connections such as type, bandwidth, speed, latency, and so on are specified in ns-2 by the network .tcl architecture description.

During the simulation, data is generated by the Matlab models and published to the RTI according to the HLA standard. The NCSWT federate subscribes to the data, and passes it via sockets to traffic generator objects in ns-2, which insert the data into the ns-2 network simulation. The traffic generator objects collect received packets of data during the ns-2 simulation and pass the data back to the NCSWT federate interface via sockets. The NCSWT federate publishes the data to the RTI so the Matlab federates can subscribe to the data.

The traffic generator class in ns-2 is designed to create fictitious traffic for network simulations, but we modified it to inject real data received from the NCSWT federate. The modified traffic generator also tests to see if it has received data from any other agents within ns-2, and it passes any received data back to the NCSWT federate interface to be published to the RTI and picked up by the appropriate federates. The traffic generator object is scheduled to be called more often than messages can arrive to ensure that additional delay is not added. In our simulations we schedule the traffic generator to be called at a rate of at least ten times faster than the smallest sampling rate.

3.5 Instrumentation and Monitoring

Monitoring and analyzing the results of NCS experiments is important to further the understanding of the systems. Our framework allows experimental results to be visualized using the tools available in MATLAB or ns-2.

MATLAB provides an extensive toolbox of visualization and monitoring that can be leveraged to display the results of experiments. These tools allow the user to plot and perform statistical analysis on any aspect of the simulation within the MATLAB federates. Further, a MATLAB federate can be designed to subscribe to all published data in the RTI to monitor any variables passed, so it can be used as an entire system monitor for debugging or analyzing a scenario.

The simulation can also be evaluated using tools provided with ns-2. In ns-2, an animation tool called nam can be used to view network simulation traces. The tool supports topology layout, packet level animation, and various data inspection tools. Moreover, other network results, e.g. packet delay and loss, can be obtained by analyzing the existing traces in ns-2. Ns-2 generates trace files automatically during simulation, which contain all the timing information of when packets are sent and received. For example, the end-to-end packet delay can be computed by subtracting the receive time and the send time of the packets. Also the overall end-to-end packet loss rate can be computed by dividing the total number of received packets by the total number of sent packets.

4. CASE STUDIES

The NCSWT tool presented in this work is intended to provide users with a flexible, extensible mechanism for simulating realistic NCS. We present simulation scenarios that demonstrate the validity of our approach as well as show how realistic network effects such as time-varying delays and packet losses are handled and how they affect the overall performance of the system.

4.1 Simple Networked Control System

The goal of this case study is to evaluate the simulation accuracy of the NCSWT. To achieve this goal, we implement a simple NCS solely in ns-2 and the NCSWT, and compare the simulation results.

To evaluate the accuracy of the NCSWT system, we simulate an NCS composed of the robotic arm and a digital controller using the NCSWT. The digital controller runs at a sampling time of 0.1s. The reference velocity profile is a sine wave with a period of 20s and an amplitude of 1 meters per second. The plant and controller dynamical models use a passive control architecture and are designed in Matlab/Simulink.

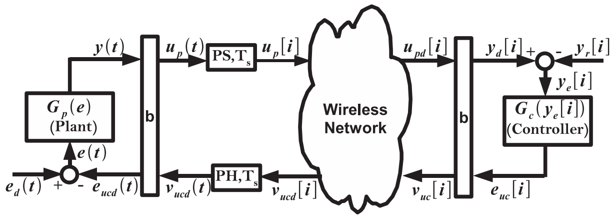


Figure 4: Passivity Based Control Architecture for a Single Plant System and a Single Controller Over Wireless Networks

Figure 4 depicts a passive control architecture for the digital control of a continuous plant system, over a wireless network. In [12], the architecture is shown to be passive by design, which means it ensures stability of the NCS in the presence of network uncertainties such as time varying delays and packet losses. For this case study, the plant system in Figure 4 is a single degree of freedom robotic arm. The plant system takes torque control command as input and outputs velocity. The robotic arm is modeled as a single-input-single-output continuous linear-time invariant system and can be represented in the following state-space form

$$\dot{x} = Ax(t) + Be(t)$$

$$y(t) = Cx(t) + De(t)$$

where $x \in \mathbb{R}$ is the state, $u \in \mathbb{R}$ is the input control command, and $y \in \mathbb{R}$ is the velocity output of the plant. The matrices A and B define the state matrices while the matrices C and D define the output matrices.

The controller sends control commands to the plant to track a desired velocity profile. The controller is modeled as a single-input single-output discrete linear time invariant system and similar to the plant, it can be represented in state-space form. The controller and the plant communicate through a UDP connection over an IEEE 802.11-based wireless network.

For comparison we also simulated the same NCS solely in ns-2. To accomplish this, C++ modules are developed and configured to represent the dynamics of the plant and controller. Figure 5 shows the velocity output plots from the NCSWT simulation as well as the velocity output from the ns-2 simulation. From the plot, one can see that the simulation results from the NCSWT and the results from ns-2 are almost identical indicating that the simulation tools are consistent.

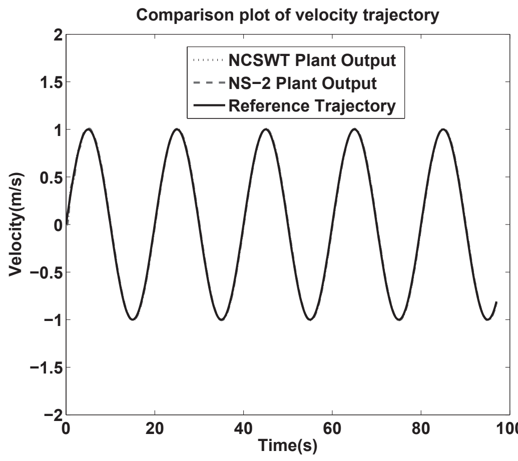
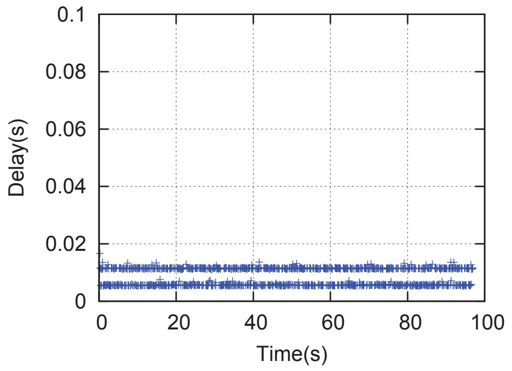
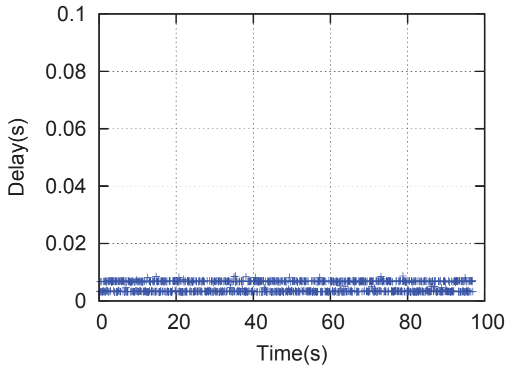


Figure 5: Comparison plot of velocity trajectory.

Figure 6 shows the end-to-end delay plots from the NCSWT and ns-2 experiments respectively. The two plots depict the delay for every packet sent for each experiment and are similar, but the the NCSWT plot shows a noticeable offset, which is directly linked to the scheduling of the ns-2 traffic generator. The traffic generator checks for received data periodically every 0.002 seconds and this causes the additional delay observed in the NCSWT end-to-end delay plot. This observed delay does not affect the performance of the plant that is being controlled since the sampling time of the traffic generator is much smaller than the smallest sampling time of the components of the NCS.



(a) Delay from the NCSWT.



(b) Delay from ns-2.

Figure 6: End-to-End delay plots .

4.2 Networked Unmanned Aerial Vehicle(UAV) System

The main objective of this case study is to evaluate the effects of various network phenomena. We use the same passivity-based architecture shown in Figure 4 but instead of the robotic arm system we use a more complex plant system: an unmanned aerial vehicle (UAV) model with an on-board controller as described in [14]. The controller is designed to cause the UAV to track a desired trajectory, and the plant receives control commands from the controller and generates x and y positions for the system. The UAV and the digital controller are modeled in Matlab/Simulink.

Experiment 1: Nominal Case
First we simulate the NCS composed of the UAV and a digital controller communicating through a single hop wireless network. The UAV (plant) and digital controller are located within the transmission range of each other, and no other network phenomena are introduced. The sampling period of the digital controller is 0.1 second. Figure 7 shows a plot of the UAV x and y position as well as the reference trajectory from the controller. The UAV position so closely tracks the reference trajectory that the difference is imperceptible in the figure.

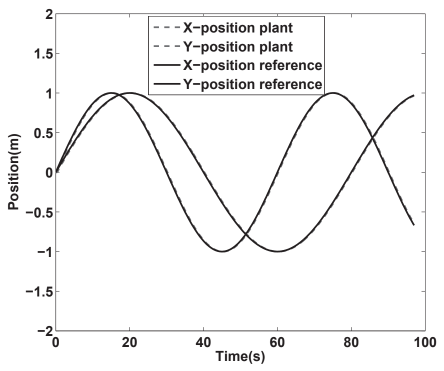


Figure 7: Plant Output for Nominal Case.

Experiment 2: Impact of packet losses
NCS using wireless connections can be unreliable, so this experiment demonstrates the performance of the UAV plant and controller model when packet loss exists on the communication network. The results for the error rates of 20% and 40% are presented in Figure 8. As the loss rate increases, the plant outputs stray further from the reference trajectories, which is expected for large loss rates.

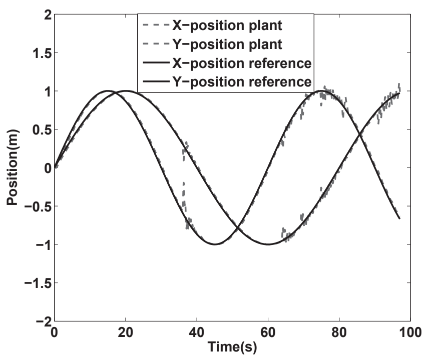
Experiment 3: Multi-hop communication
Direct connection in wireless networks requires two nodes to be within transmission range of each other; however, this may not always be the case. To allow communication between the nodes, intermediate nodes can provide relay to route the packets. Such network architectures are called wireless multi-hop networks.

We use a chain topology to test the multi-hop scenario. In this topology, nodes are formed in a chain structure with a fixed distance of 200m between neighboring nodes. The plant and the controller are located at the edges of the network. Figure 9 shows the simulation results for three and five hop chain networks. From Figure 9 it can be seen that as the number of hops increases, the plant outputs deviate farther away from the reference trajectory due to the increased delay created by the hops between the plant and the controller.

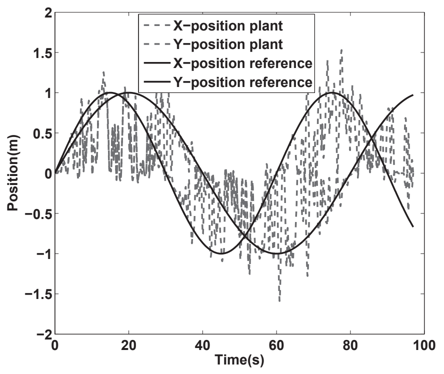
4.3 Multi-agent Networked Control System

The goal of this case study is to demonstrate the scalability and flexibility of the NCSWT for use with larger systems. Realistic NCS can be composed of a large number of components that share network resources and may operate at different sampling rates. Therefore, simulation tools must be able to handle multiple systems and appropriately handle different (and potentially varying) sampling rates. Multiple NCS agents with different sampling rates can introduce complicated resource contention in the communication channel, so it is important to accurately handle this problem to avoid introducing error. The HLA communication standard provides a framework within which our NCSWT simulation tool can precisely handle simulations with various sampling rates efficiently.

We present a scenario that demonstrates the modeling of multiple NCS. Three models of the UAV plant and controller as described in Section 4.2 are used operating with different sampling times. The three digital controllers operate at the sampling times of 0.1s, 0.15s and 0.25s respectively. The plant and controller models are designed in Matlab/Simulink. We consider two experiments: a baseline nominal case, and a case with background traffic in the network.



(a) Output plot for 20% loss rate.

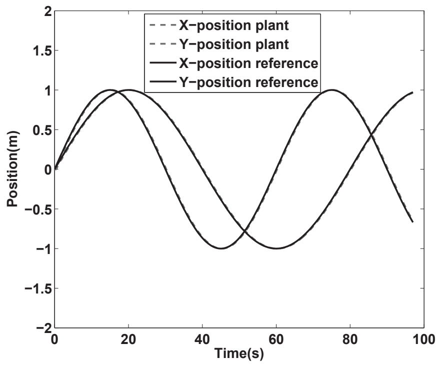


(b) Output plot for 40% loss rate.

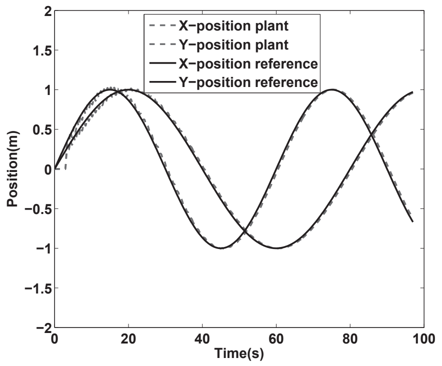
Figure 8: Plots of plant output for various uniform loss rates .

Experiment 1: Nominal Case
In this experiment the three NCS are operating in a nominal network condition without introducing additional data dropouts or losses. All six nodes are use the same network channel. The reference trajectories for this experiment were essentially identical to the reference trajectories in the single UAV system shown in Figure 7. Figure 10 shows the end-to-end delay plots. The horizontal red line in each plot indicates the sampling time for each plant-controller pair. The maximum end-to-end delay for all three plant controller pairs is less that 0.04s. The noticeable difference in plot density of NCS systems is due to the difference in their sampling times.

Experiment 2: Case with Background traffic
Since realistic NCS may not exist on a simple, dedicated network we demonstrate the incorporation of background traffic to the wireless network. Figure 11 shows a plot of the outputs of the plant showing the effect of the background traffic. It can be seen from the plots that the background traffic affects the output of the plant significantly at some times.

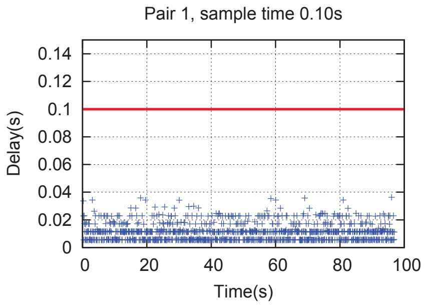


(a) Output plot for Three hops.

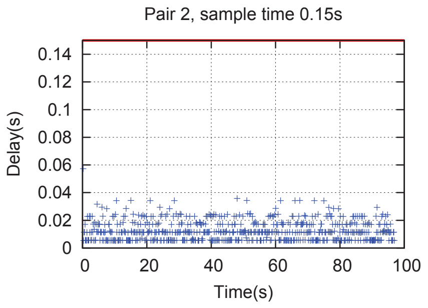


(b) Output plot for Five hops.

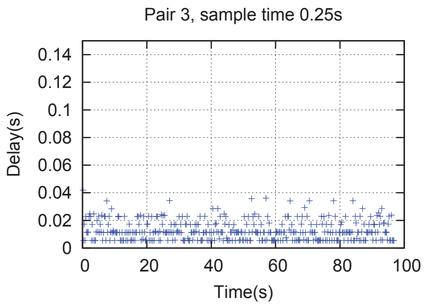
Figure 9: Plant output for multi-hop topologies .



(a) Delay plot for P1-C1 pair



(b) Delay plot for P2-C2 pair



(c) Delay plot for P3-C3 pair

Figure 10: End-to-End delay plot for multi-rate agents' nominal case scenario.

Because of the level of detail captured by the NCSWT, the conditions that cause these deviations can be closely studied and better understood.

Figure 12 presents the network delay for packets transmitted between the plant and the controller. Similar to the delay plots in Figure 10, the horizontal red line in each plot indicates the sampling time for each plant-controller pair. The figure shows similar delay between for most packets in the system, and this is reasonable since all the six nodes share the same network channel.

5. EVALUATION

The NCSWT tool provides a convenient approach for designing and simulating NCS. It solves complexity and efficiency issues faced by some stand-alone simulators by leveraging the advantages of the integrated simulation tools. The formal time synchronization mechanisms reduce the overall performance of the tool; however, we demonstrate that this performance degradation is acceptable.

5.1 Design Comparison

In order to demonstrate the convenience of our tool, we compare the ease of designing an NCS using the NCSWT to designing the same NCS using only ns-2. We use two NCS examples to demonstrate the convenience of using our tool.

Example 1: Simple NCS Model The first example demonstrates the design of the simple NCS described in the case study provided in Section 4.1. To model the system in ns-2 alone, the Matlab models of the plant and controller must be re-modeled using C++ and inserted into ns-2. The robotic arm and the controller can be easily represented as differential and difference equations respectively to accomplish this.

Figure 13 shows the implementation of the plant subsystem using Matlab/Simulink, and Figure 14 shows some code from the ns-2 implementation of the plant subsystem using C++. In the Matlab/Simulink model the plant is modeled using the continuous state-space block of the built-in Simulink libraries. The passive sampler, passive hold, and wave transform blocks are modeled using blocks from a passivity based control library. The ns-2 standalone implementation utilizes C++ modules to represent the plant. Since ns-2 cannot model the actual continuous dynamics of the plant, a discrete approximation of the plant is used. The passive sampler, passive hold, and wave transform blocks are implemented as C++ code as well.

Example 2: Complex NCS Model In the second example, we use a complex NCS model to demonstrate the flexibility and convenience of our tool. Figure 15 shows the Matlab/simulink model of plant describing the UAV with an on-board controller. This is the same plant used for the case study in Section 4.2. In this model, integrator and saturation blocks from the simulink library are used to accurately simulate the continuous dynamics of the UAV. By using Matlab/Simulink in our approach we can accurately realize the UAV behavior.

Implementing this system exclusively in ns-2 can result in inaccurate system behaviors and complicated script implementation. This is because ns-2 is based on discrete event model of computation and can not easily realize the continuous behavior of the UAV without significant modifications, which will impact performance. The UAV system is sensitive to model uncertainties and since ns-2 cannot accurately realize the behavior of the integrators, modeling this system in ns-2 will not accurately capture the behavior of the overall system. Such complex models highlight the strength of our tool, which leverages the desirable features of both Matlab/Simulink and ns-2 and avoids the difficulties faced by implementing an NCS in one of the simulators alone.

5.2 Performance

Trade offs exist when attempting to simulate realistic systems efficiently, so in this work we focus on generating accurate simulations, and therefore, simulation efficiency is reduced. Additional overhead in terms of the running time exists in the NCSWT to enforce the time syncronization using the HLA standards; however, the performance of the tool is still good.

The run-time efficiency of the experiments for the networked UAV system presented in Section 4.2 (denoted SANCS) is shown in Table 1. The information in the table shows how long it takes to complete a 98-second simulation using our tool under the various network conditions considered in Section 4.2. It can be seen that the performance of the tool depends on the network complexity.

Table 1: Run-time efficiency (SANCS)

Scenarios		**Actual Duration (in minutes)**
Nominal		58.6
Losses	20%	61.5
	30%	65.0
	40%	66.0
Background Traffic	1 pair	71.6
	3 pairs	87.2
Multi-hop Network	2 hops	65.8
	3 hops	78.9
	4 hops	83.7
	5 hops	87.0

Table 2 presents the run-time efficiency for the Multi-Agent NCS case study (denoted MANCS) from Section 4.3. It can be seen that our tool is able to generate results in under two hours using two standard desktop computers connected via a standard Ethernet subnet. While this may not be as fast as other simulators, the accuracy is far superior. Further performance improvements are possible through software and hardware improvements, and will be the focus of future work.

Table 2: Run-time efficiency (MANCS)

Scenarios	**Actual Duration (in minutes)**
Nominal	66.8
Background Traffic	83.3

6. CONCLUSION

Analysis of NCS is a critical task because the intricate nuances of the models can behave in unexpected ways when subjected to real world conditions. Typically NCS systems are highly complex and are difficult to model and analyze. In this work we present the NCSWT, an NCS evaluation tool based on HLA that can be used to model and simulate realistic NCS using an underlying framework for time synchronization and data passing. The tool provides integration mechanisms for control system models in Matlab/Simlink and network models in ns-2. It also provides rapid prototyping integration as well as run-time support to coordinate simulations. We demonstrate the power of the tool by presenting case studies of models with various network conditions and control system parameters, such as sampling rates. We also present the performance of the tool and compare it to other approaches to demonstrate its power and flexibility. In the future we would like to simulate larger, more complex models with the NCSWT as well as improve the overall performance of the tool.

7. ACKNOWLEDGEMENTS

We want to thank the referees and editors for their extremely useful suggestions. This work was supported in part by the National Science Foundation (Awards CNS-1035655 and CCF-0820088), as well as the U.S. Army Research Office (ARO W911NF-10-1-0005), and Lockheed Martin.

8. REFERENCES

[1] A. Al-Hammouri, D. Agarwal, V. Liberatore, H. Al-Omari, Z. Al-Qudah, and M. S. Branicky. Demo abstract: a co-simulation platform for actuator networks. *Sensys*, 2007.

[2] J. Baillieul and P. Antsaklis. Control and communication challenges in networked control systems. *Proceedings of the IEEE*, 95(1):9 – 28, 2007.

[3] A. Cervin, M. Ohlin, and D. Henriksson. Simulation of networked control systems using truetime. *In Proc. 3rd International Workshop on Networked Control Systems: Tolerant to Faults*, 2007.

[4] O. Community. Omnet++ discrete event simulation system. *Home page: http://www.omnetpp.org*, 2004.

[5] M. Emerson and J. Sztipanovits. Techniques for metamodel composition. In *OOPSLA - 6th Workshop on Domain Specific Modeling*, pages 123–139, 2006.

[6] M. S. Hasan, H. Yu, A. Griffiths, and T. C. Yang. Co-simulation framework for network control systems over multi-hop mobile adhoc networks. *In Proc. 17th International Federation of Automatic Control World Congress*, 2008.

[7] U. Hatnik and S. Altmann. Using modelsim, matlab/simulink and ns for simulation of distributed systems. *Parallel Computing in Electrical Engineering, International Conference on*, 0:114–119, 2004.

[8] HLA. Ieee standard for modeling and simulation (m amp;s) high level architecture (hla) - framework and rules. *IEEE Std. 1516-2000*, pages i –22, 2000.

[9] G. W. Irwin, J. Colandairaj, and W. G. Scanlon. An overview of wireless networks in control and monitoring. *In Proc. International Conference on Intelligent Computing (ICIC)*, pages 16–19, 2006.

[10] G. Karsai, A. Ledeczi, S. Neema, and J. Sztipanovits. The model-integrated computing toolsuite: Metaprogrammable tools for embedded control system design. In *IEEE Joint Conference CCA, ISIC and CACSD*, 2006.

[11] T. Kohtamaki, M. Pohjola, J. Brand, and L. Eriksson. Piccsim toolchain - design, simulation, and automatic implementation of wireless networked control systems. In *IEEE Conference on Networking, Sensing, and Control*, 2009.

[12] N. Kottenstette, X. Koutsoukos, J. Hall, P. Antsaklis, and J. Sztinapovits. Passivity-based design of wireless networked control systems for robustness to time-varying delays. In *29th EEE Real-Time Systems Symposium (RTSS 2008)*, 2008.

[13] F. Kuhl, R. Weatherly, and J. Dahmann. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Prentice Hall, 1999.

[14] H. LeBlanc, E. Eyisi, N. Kottenstette, X. Koutsoukos, and J. Sztipanovits. A passivity-based approach to deployment in multi-agent networks. *International Conference on Informatics in Control, Automation and Robotics*, 2010.

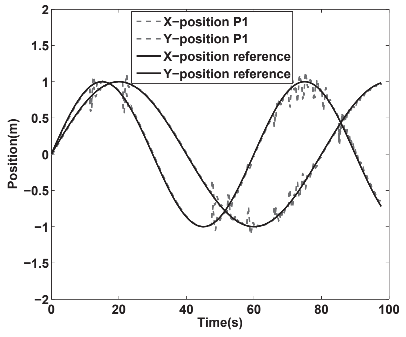
[15] I. T. MathWorks. Simulink. *Dynamic System Simulation for MATLAB, Version 7.1*, 2008.

[16] H. Neema, G. Hemingway, J. Green, B. Williams, J. Sztipanovits, and G. Karsai. Rapid synthesis hla-based heterogeneous simulation: A model-based integration approach. Technical report, ISIS/Vanderbilt University, 2008.

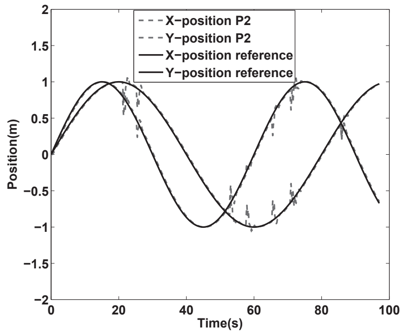
[17] S. Neema, T. Bapty, X. Koutsoukos, H. Neema, J. Sztipanovits, and G. Karsai. Model based integration and experimentation of information fusion and c2 systems. In *The 12th International Conference on Information Fusion*, 2009.

[18] T. Patki, H. Al-Helal, J. Gulotta, J. Hansen, and J. Sprinkle. Using integrative modeling for advanced heterogeneous system simulation. In *International Conference and Workshop on the Engineering of Computer Based Systems*, 2009.

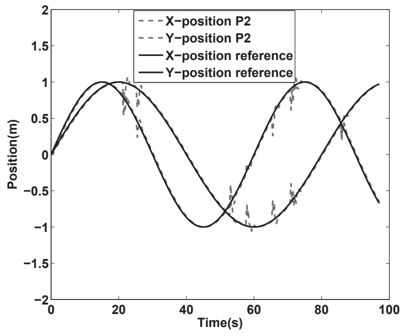
[19] T. V. Project. The network simulator ns-2. *Home page: http://www.isi.edu/nsnam/ns/index.html*, 2004.



(a) Output plot for Plant P1.

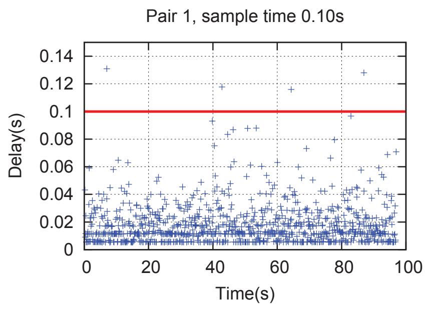


(b) Output plot for Plant P2.

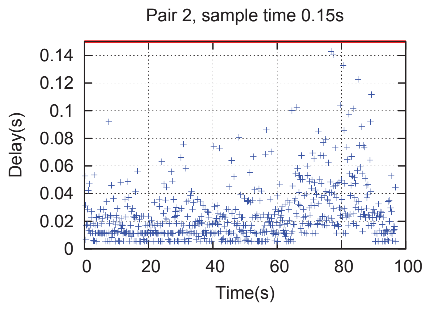


(c) Output plot for Plant P3.

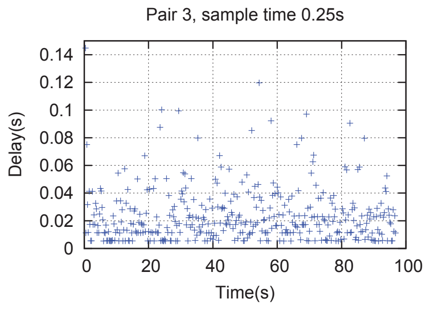
Figure 11: Output plots for the addition of background traffic in multi-rate agents network.



(a) Delay plot for P1-C1 pair

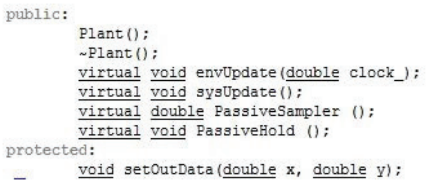


(b) Delay plot for P2-C2 pair



(c) Delay plot for P3-C3 pair

Figure 12: End-to-End delay for the addition of background traffic in multi-rate agents network.



```
public:
        Plant();
        ~Plant();
        virtual void envUpdate(double clock_);
        virtual void sysUpdate();
        virtual double PassiveSampler ();
        virtual void PassiveHold ();
protected:
        void setOutData(double x, double y);
```

Figure 14: Snapshot of ns-2 implementation of simple plant node

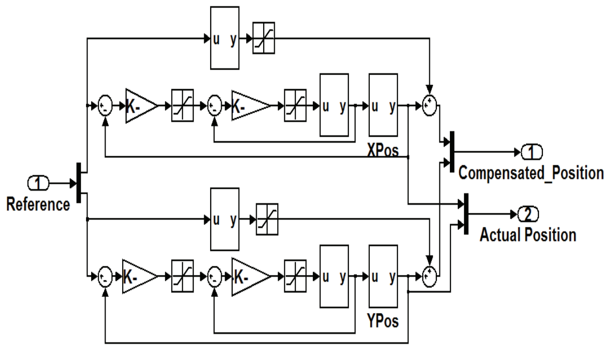


Figure 15: Simulink model of complex plant example.

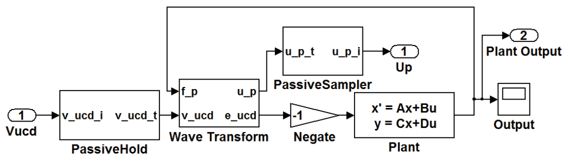


Figure 13: Simulink model of simple plant node.