NCSWT: An Integrated Modeling and Simulation Tool for Networked Control Systems

Emeka Eyisi*, Jia Bai*, Derek Riley**, Jiannian Weng*, Yan Wei*, Yuan Xue*, Xenofon Koutsoukos* and Janos Szipanovits*

*Institute for Software Integrated Systems
EECS Department
Vanderbilt University
Nashville, TN, USA

**Department of Computer Science
University of Wisconsin-Parkside
Kenosha, WI, USA

ABSTRACT

This paper presents the Networked Control Systems Windtunnel (NCSWT), an integrated modeling and simulation tool for the evaluation of networked control systems (NCS). NCSWT integrates Matlab/Simulink and ns-2 using the High Level Architecture (HLA). Our implementation of the NCSWT based on HLA guarantees accurate time synchronization and data communication in heterogenous simulations. NCSWT uses the Model Integrated Computing (MIC) techniques to define HLA-based model constructs such as federates representing the simulators and interactions between the simulators. NCSWT also uses MIC techniques to define models representing the control system and network dynamics for the rapid synthesis of simulations.

Categories and Subject Descriptors

I.6.7 [**SIMULATION AND MODELING**]: Simulation Support Systems—*Environments*

General Terms

Design, Experimentation

Keywords

Modeling, Simulation, Networked Control Systems, HLA

1. INTRODUCTION

Networked control systems (NCS) have gained increasing attention in recent years due to their cost effective and flexible applications [6]. NCS are often employed in critical settings, therefore the assurance of properties such as stability, performance, safety and security are essential. Currently, many NCS are designed without considering the effects of the network operating environment (e.g time-varying delays and packet losses). Such limitations in the system design phase can lead to catastrophic consequences when the actual systems are deployed as the overall system behavior depends on network dynamics and uncertainties.

HSCC'12, April 17–19, 2012, Beijing, China.

As NCS become increasingly complex, it becomes more challenging to formally analyze their performance, stability, safety and security properties. As a result, there is a pressing need to evaluate both the control and network components of NCS together for a rapidly growing number of applications, such as unmanned aerial vehicles (UAVs) and industrial control systems. Simulation is a powerful technique for evaluation and can be used at various design stages, but it requires the support of appropriate tools during both the design-time and run-time stages in order for the process to be efficient and less prone to errors.

Currently, several simulators have been used for NCS but have limited capabilites. For example, Matlab/Simulink is a popular tool for the evaluation control systems [3]. Although network simulation is provided in Matlab/Simulink using toolboxes such as TrueTime [7], the accuracy of the simulation depends on the level of abstraction of the network protocol models. Specifically, TrueTime only supports link layer protocols but not higher level protocols such as TCP or UDP protocols, which are essential for simulating the communication network of a NCS. Packet-level network simulators such as ns-2 [2], provide a detailed implementation of the network stack for packet level data transmission. Yet, using only ns-2 for NCS evaluation requires the control algorithm to be fully implmented in a high-level language such as C++. This becomes very difficult as the complexity of the NCS increases. In order to develop a realistic and accurate simulation of NCS, we need a modeling and simulation environment that can integrate existing tools for the simulation of the control dynamics as well as the networking system of a NCS.

The integration of existing tools for the simulation of NCS, although very beneficial, faces several challenges. The first challenge is the design-time scalability of modeling NCS. This involves the ability to rapidly design and model NCS of various complexity and size. The second challenge is time synchronization of the heterogenous simulation components during execution. Given that the simulators operate in potentially different time scales using disparate time models, time synchronization between the simulators is critical to preserve the correctness of the simulation. The third challenge involves the data communication between the simulators to ensure consistent data semantics during the simulation. Finally, the fourth challenge involves the run-time scalability which is the ability of the simulation environment to handle the simulation of large and complex NCS.

In order to address these challenges, we present an integrated modeling and simulation tool for NCS, called the Networked Control Systems Wind Tunnel (NCSWT) [1], which combines the network simulation capabilities of ns-2 with the control design and simulation capabilities of Matlab/Simulink. NCSWT addresses the

challenge of design-time scalability of modeling NCS by adopting the Model Integrated Computing (MIC) techniques [12]. MIC is an approach for the development of complex software systems, applicable in all phases of system design and maintenance. The key idea in MIC is to create Domain-Specific Modeling Languages (DSMLs) using a meta-modeling framework, and then, describe objects in terms of the domain-specific models. We present three DSMLs which abstract a NCS simulation from three design views at two levels: at the low level are the simulation models for the control subsystem and the networking subsystem of a NCS, defined by the the Control Design Modeling Language (CDML) and the Network Design Modeling Language (NDML) respectively; at the high level is the NCS integration model, based on the High Level Architecture (HLA) standards, defined by the NCSWT model integration language (NCSWT MIL). The DSMLs, developed using the Generic Modeling Environment (GME) [15], facilitate the rapid design and modeling of NCS. The DSMLs are designed to ensure the consistency of data semantics among the simulators used in the simulation of a NCS.

NCSWT addresses the challenges involving time synchronization and data communication by adopting the High Level Architecture (HLA) for the implementation of the run-time simulation environment [14]. HLA is a standard for simulation interoperability that allows independently developed simulations, each designed for a particular problem domain, to be combined into a larger and more complex simulation. In HLA, the independent simulators are known as federates and the larger simulation formed by the interconnection of the federates is known as the federation. The HLA standard provides a set of services to accurately handle time management and data distribution among the independent simulators. NCSWT utilizes the time management services provided by the HLA to ensure that the time model in the control system simulated in Matlab/Simulink and the time model in the networking system simulated in ns-2 are synchronized. NCSWT also utilizes the data distribution services to ensure the correct exchange of data between the simulations of the control dynamics and communication network of a NCS.

Finally, we demonstrate the NCSWT tool through an evaluation in Section 4. We list the required software packages for NCSWT tool and discuss the design-time efficiency and run-time efficiency for a specific NCS case study.

2. RELATED WORK

Several efforts have been made towards integrating multiple simulators in order to effectively simulate NCS. A tool chain PiccSIM was developed in [13], that allows the integration of Matlab/Simulink models with ns-2. PiccSIM also provides a graphical user interface for the design of networked control systems and the automatic code generation of ns-2 and Matlab/Simulink models. In [9], a special simulator coupling, implemented in C/C++ is used to integrate the simulators, ModelSim, Matlab/Simulink and ns-2 to establish the communication between the simulators. Other tool integration projects also targeted for NCS include [10] [5] [8] and references therein.

NCSWT differs from these other simulation tools for NCS in multiple aspects. First, our integration of Matlab/Simulink and ns-2 for the simulation of NCS is based on the HLA standard, and hence, ensures a correct and valid NCS simulation. Secondly, our model-based approach provides a clear model of NCS architecture that tightly integrates the control design and communication network in NCS providing a well-defined abstraction of the information exchange between the two subsystems. Such a design-time modeling environment that supports NCS integration is not available with existing tools. As a result, the interactions between the control system and networking components are described in an ad-hoc manner, resulting in possibly error-prone designs. Finally, the design-time efficiency and automatic code generation based on DSMLS is a strong feature of our tool.

In [16], we presented a preliminary version of NCSWT which is substantially different from the current version. First, the early work in [16] is a pure run-time simulation environment while the current version presents an integrated modeling and simulation tool suite. Second, the current version has a new run-time environment implementation based on only Linux compared with the Linux-Windows-based implementation presented in [16]. This revision eliminates the need of using TCP sockets to perform a proprietary communication protocol between the two simulators and significantly improves the run-time efficiency.

3. NCSWT

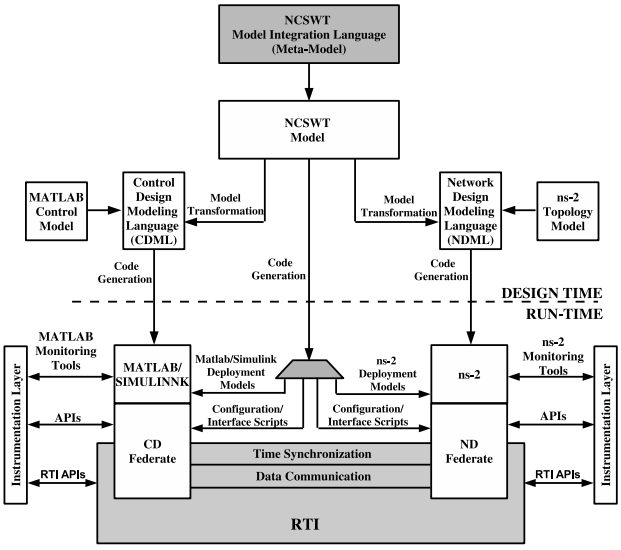

Figure 1: Overview of NCSWT

Figure. 1 shows an overview of the NCSWT tool architecture. The architecture is composed of two main parts, the design-time models and the run-time components. We provide a brief description of NCSWT, a more detailed description of the tool along with extensive experimental validations and results from case studies are provided in [1].

3.1 Design-Time Models

In Figure. 1, the design-time models are used to define the NCS and its components in order to facilitate the simulation of a NCS at run-time. The design-time models are defined by three DSMLs.

3.1.1 NCSWT Model Integration Language (NCSWT MIL)

The NCSWT MIL specifies the NCS in terms of HLA-based constructs, such as federates representing the simulators for each of the components of the NCS and interactions representing the communication between the simulators. A model created using NCSWT MIL is refered to as the base architecture of the NCS. From the base architecture model, the executables for configuring the run-time environment for a specific NCS are generated. The NCSWT MIL is an extension of the work in [11] which introduces

a DSML for HLA-based simulations. The NCSWT MIL describes the tight coupling between the control design and the networking subsystems of the NCS by defining how the two subsystems interact. Two types of federates can be modeled in the NCSWT MIL, the *CDFederate* and the *NDFederate*. The *CDFederate* models an instance of the Matlab/Simulink simulator for each corresponding control design component of NCS while the *NDFederate* models an instance of the ns-2 simulator for simulating the communication network of NCS.

Three type of interactions can be modeled in the NCSWT MIL to represent information exchange in NCS, *NetworkInteraction*, *CrossLayerInteraction* and *ControlDesignInteraction*. The *NetworkInteraction* models the exchange of packets over the communication network. The *CrossLayerInteraction* models the information exchange between the network and application layers of a network protocol stack and the *ControlDesignInteraction* models the information exchanged between components of the control system that are transmitted or received, by other means other than the communication network.

3.1.2 Control Design Modeling Language (CDML)

The CDML defines the modeling concepts for specifying the dynamic behavior of the control design components. These include the dynamics of the plant (system to be controlled) and the digital controller. A model created by CDML is a refinement of the base architecture model of a NCS, created in the NCSWT MIL, with the details regarding the dynamics of the control system added. In order to maintain consistency with the base architecture model defined in the NCSWT MIL, a model transformation is used to transform the base architecture model to a model in CDML. Then the control design concepts in CDML are used to specify the dynamics of the components in the NCS. In CDML, using a set of modeling primitives such as T_s (Sampling Time), ModelName, ModelLibraryName etc., a user can specify the model and parameters that define the dynamic behavior of a control system component. Using the defined parameters in CDML, executable Matlab/Simulink code can be generated for implementing the control system.

3.1.3 Network Design Modeling Language (NDML)

The NDML defines the modeling concepts for specifying the dynamics of the communication network. This includes the capacity, loss rate models, routing and other additional properties to realize a communication network. Similar to CDML, a model transformation is used to transform the base architecture model to a model in NDML. Then the concepts defined in NDML are used to specify the network properties for the NCS. A user can specify the transport agent, loss model of network links and other various network properties to simulate the network dynamics. Run-time network configuration and model scripts can then be generated based on the defined parameters for deployment on ns-2.

3.2 Run-Time Components

In Figure. 1, the run-time components represent the main software components and interfaces for the actual realization of a simulation using the HLA framework. These components include the Run-Time infrastructure (RTI), the federates, and all the necessary configuration and glue code for the interfaces as well as monitoring tools for visualizing and evaluating the results.

3.2.1 Run-Time Infrastructure (RTI) and Federates

The RTI, an implmentation of the HLA standard, manages the coordination of time and data passed between federates. Using interactions, federates communicate between each other through the RTI. A number of commercial and academic RTI implementations are available. Currently, we use Portico 1.0.2, an open source cross-platform HLA implementation which supports both C++ and Java clients [4].

Each federate represents a single instance of the corresponding simulator's interface to the RTI. For example, the *NDFederate* is a software component that interfaces the ns-2 simulator with the RTI.

We briefly describe the NCSWT run-time services provided by the RTI.

(a) Time Synchronization: In a HLA-based federation, each federate has its own logical time. The RTI preserves the causality of the federation by ensuring that no simulation receives an event that occured in the past relative to its own. The RTI ensures the accurate progression of time through the use of a time advance grant request (TAR) and time advance grant (TAG) mechanism [16]. For ns-2, this mechanism is integrated into the ns-2 scheduler while in Matlab/Simulink, the mechanism is integrated as part of the interface code to the Matlab federate.

(b) Data Communication: The RTI uses a publish-and subscribe mechanism for passing messages through the federation in order to ensure the consistent data communication and coordination between the federates [16]. The type of messages exchanged between the federates are defined by the interactions modeled in the NCSWT MIL during the design-time and is integrated in the generated code deployed during the run-time simulation.

4. EVALUATION

The simulation of a NCS using the NCSWT tool requires two major steps. The first step involves the modeling of NCS using the three DSMLs discussed in Section 3.1, followed by the generation of all the necessary models, configurations and glue code for the simulation of the NCS. This step is performed on a computer running a Windows operating system. The second step involves the deployment of the generated models, configurations, and glue code followed by the execution of the simulation. This step is performed on a computer running a Linux operating system.

The NCSWT tool requires the software packages as shown in Table 1. Matlab/Simulink and ns-2 are used for the simulation of the control design and networking subsystems of the NCS respectively. Portico 1.0.2 is the RTI implementation of the HLA used for running the federation. GME is the graphical environment used for the modeling and generation of all the necessary components for the simulation of the NCS. UDM is utilized in the model transformations from NCSWT MIL to CDML and NDML. Microsoft Visual Studio is used for the execution of the code generators and model transformations from the three DSMLs. Eclipse is used for the compilation of the run-time components required for the simulation. We have evaluated the NCSWT tool using various NCS case studies some of which include a NCS composed of a single plant

Table 1: Required Software Packages

1. Matlab/Simulink, `www.mathworks.com`
2. ns-2, `http://isi.edu/nsnam/ns`
3. Portico 1.0.2, `www.porticoproject.org`
4. Generic Modeling Environment (GME), `www.isis.vanderbilt.edu/Projects/gme`
4. Universal Data Model (UDM), `www.isis.vanderbilt.edu/tools/UDM`
5. Microsoft Visual Studio 2008 or later, `www.microsoft.com/visualstudio`
6. Eclipse, `www.eclipse.org`

and controller and a NCS composed of multiple plants and multiple controllers with asynchronous sampling times [16].

We present some results for a single linear continuous-time plant and single linear discrete-time controller NCS. This NCS involves the digital control of an unmanned aerial vehicle (UAV), representing the plant, over a 802.11b wireless network to track a desired trajectory. In order to provide insights about the efficiency of using the NCSWT tool, we present the design-time efficiency and the run-time efficiency. For the design-time efficiency, we consider the amount of code that is automatically generated for simulating the NCS. Table 2 provides a summary of the size of code and models that are automatically generated from the design-time models. For the run-time efficiency, we consider the actual time it takes to simulate the NCS. Table 3 shows the time durations for simulating the NCS in various multi-hop network topologies and in the presence of network uncertainties such as packet loss and time varying delays. The time durations shown in the table are the actual times required to run 100 seconds of logical simulation time.

Table 2: Generated Code for NCS Example

Files	Size
1. Matlab models	100 Kilobytes
2. Matlab glue code	132 Kilobytes
3. ns-2 model and topology scripts	20 Kilobytes
4. ns-2 glue code	160 Kilobytes
5. Federation startup script	4 Kilobytes

Table 3: Time Efficiency for NCS Example

Scenarios		Actual Duration (in minutes)
Nominal		5.5
Packet Losses	20%	8.4
	30%	11.4
	40%	13.5
Multi-hop Network	3 hops	17.4
	4 hops	22.2
	5 hops	26.2

5. CONCLUSION

We presented the integration tool, NCSWT for the modeling and simulation of networked control systems. We described the HLA-based approach guiding the tool's implementation as well as the MIC techniques for the rapid synthesis of components required for the simulation of a NCS. Additionally, we provided an evaluation of tool.

6. ACKNOWLEDGEMENT

This work is supported in part by the U.S. Army Research Office (ARO W911NF-10-1-0005), the National Science Foundation (CNS-1035655, CCF-0820088) and Lockheed Martin. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government.

7. REFERENCES

[1] NCSWT: An Integrated Modeling and Simulation Tool for Networked Control Systems. `http://vanets.vuse.vanderbilt.edu/dokuwiki/doku.php?id=research:cps`.

[2] The Network Simulator ns-2. `http://isi.edu/nsnam/ns/`, 2004.

[3] MATLAB, The Language of Technical Computing. `http://www.mathworks.com`, 2008.

[4] Portico RTI. `http://www.porticoproject.org`, 2010.

[5] A. Al-Hammouri, M. Branicky, and V. Liberatore. Co-simulation Tools for Networked Control Systems. *Hybrid Systems Computation and Control, Lecture Notes in Computer Science*, 4981:16–29, 2008.

[6] P. Antsaklis and J. Baillieul. Special Issue on Technology of Networked Control Systems. *Proc. of the IEEE*, 95(1):5–8, Jan. 2007.

[7] A. Cervin, M. Ohlin, and D. Henriksson. Simulation of Networked Control Systems Using TrueTime. *In Proc. 3rd Int. Wkshp. on Networked Control Systems: Tolerant to Faults*, 2007.

[8] M. Hasan, H. Yu, A. Carrington, and T. Yang. Co-simulation of wireless networked control systems over mobile ad hoc network using SIMULINK and OPNET. *IET Comm.*, 3(8):1297 –1310, Aug. 2009.

[9] U. Hatnik and S. Altmann. Using ModelSim, Matlab/Simulink and NS for Simulation of Distributed Systems. *Int. Conf. on Parallel Computing in Electrical Engineering*, 0:114–119, 2004.

[10] O. Heimlich, R. Sailer, and L. Budzisz. NMLab: A Co-simulation Framework for Matlab and NS-2. In *2010 Second Int. Conf. on Advances in System Simulation (SIMUL)*, pages 152 –157, Aug. 2010.

[11] G. Hemingway, H. Neema, H. Nine, J. Sztipanovits, and G. Karsai. Rapid synthesis of high-level architecture-based heterogeneous simulation: a model-based integration approach. *SIMULATION*, 2011.

[12] G. Karsai, J. Sztipanovits, A. Ledeczi, and T. Bapty. Model-Integrated Development of Embedded Software. *Proc. of the IEEE*, 91(1):145–164, Jan 2003.

[13] T. Kohtamaki, M. Pohjola, J. Brand, and L. Eriksson. PiccSim Toolchain - Design, Simulation, and Automatic Implementation of Wireless Networked Control Systems. In *IEEE Conf. on Networking, Sensing, and Control*, 2009.

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

[15] A. Ledeczi, M. Maroti, A. Bakay, G. Karsai, J. Garrett, C. Thomason, G. Nordstrom, J. Sprinkle, and P. Volgyesi. The Generic Modeling Environment. *Wkshp. on Intelligent Sig. Proc.*, May 2001.

[16] D. Riley, E. Eyisi, J. Bai, Y. Xue, X. Koutsoukos, and J. Sztipanovits. Networked Control System Wind Tunnel (NCSWT)- An evaluation tool for networked multi-agent systems. In *4th Int. ICST Conf. on Simulation Tools and Techniques (SIMUTools 2011),*, March 2011.