Universidad Nacional de Rosario
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Departamento de Control – CIFASIS-CONICET

Tesis Doctoral

Herramientas Integradoras para Modelado, Simulación y Control de Redes de Datos

Rodrigo Castro

Director: Prof. Dr. Ernesto Kofman
Co-Director: Prof. Dr. Gabriel Wainer
Miembros del Jurado: Prof. Dr. Pablo Jacovkis
Prof. Dr. Anibal Zanini
Prof. Dr. Pablo Lotito

Tesis presentada en la Facultad de Ciencias Exactas, Ingeniería y Agrimensura, en cumplimiento parcial de los requisitos para optar al título de

Doctor en Ingeniería

Diciembre de 2010

Certifico que el trabajo incluído en esta tesis es el resultado de tareas de investigación originales y que no ha sido presentado para optar a un título de postgrado en ninguna otra Universidad o Institución.

Rodrigo Daniel Castro

A Daniel y Adriana, mis viejos,
por haberme dado la educación y el apoyo incondicional para perseguir siempre mis sueños.

A Natalia, mi esposa,
por ser mi sueño preferido y por todo el tiempo que me regaló para poder escribir esta Tesis.

Agradecimientos

Sin dudas debo esta Tesis principalmente a mi director Ernesto Kofman, mi amigo de muchos años que además supo convertirse también en guía académico sin morir en el intento. Y a mi codirector, Gabriel Wainer, por haber comenzado como guía académico y haberse convertido en un nuevo amigo.

Toda la Tesis fue desarrollada en un triángulo geográfico: Buenos Aires (donde vivo y hago docencia), Rosario (lugar de trabajo de Ernesto) y Ottawa, Canadá (lugar de trabajo de Gabriel). Mas allá de los numerosos viajes, fue gracias a la infalible "prediposición online"de Ernesto y Gabriel que pude aprender y avanzar como si la distancia no existiese. Recibí de ambos ayudas que excedieron largamente las esperadas de sus roles.

Agradezco también a la Fundación YPF por adjudicarme a comienzos de 2007 la beca doctoral que me sirvió como sustento económico parcial para poder desarrollar la investigación de esta Tesis en la Universidad Nacional de Rosario y el Centro Internacional Franco Argentino de Ciencias de la Información y de Sistemas (CIFASIS) del CONICET.

Por medio de Ernesto tuve además la suerte de conocer a François Cellier, del ETH Zurich, Suiza. Gracias a la confianza de François al recibirme en su laboratorio durante una estadía en 2009 surgió uno de los aportes que más valoro de la Tesis, el método DQSS para resolver ecuaciones diferenciales con retardos. Gracias a mis amigos Pablo Echeverría y Tamara Tanos por abrirme las puertas de su casa en Suiza durante aquel mes, uno de los más divertidos del doctorado.

Mi gratitud hacia François excede su rol de colaborador. En su curso de modelado en Rosario durante 2007 me reveló la existencia de los modelos mundiales y del modelado por medio de flujos de emergía (embodied energy) para tratar problemas de sustentabilidad energética a nivel global. Este hecho modificó radicalmente la orientación de mis intereses científicos, sospecho que de forma irreversible.

También debo agradecer a Hugo Scolnik e Irene Loiseau por la confianza brindada. Ambos me gestionaron la oportunidad de acceder a un lugar físico de trabajo en Buenos Aires en un excelente entorno académico, el Departamento de Computación de la Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires. Allí, gracias a Gabriel y también a Ezequiel Glinsky pude dar mis primeros pasos como docente en el curso de Simulación de Eventos Discretos, para luego tomar la responsabilidad completa del curso, sin que el desafío se transforme en un molino de viento. Gracias Eze !

Gracias también a Pablo Jacovkis, por las numerosas y valiosas discusiones que me sirvieron para tomar decisiones importantes durante el doctorado, y también por enriquecer mi forma de pensar la ciencia en relación a la sociedad y a la Historia.

Quien seguramente rechazará el siguiente agradecimiento es Sergio Junco. Argumentará su nula participación en los temas desarrollados en esta Tesis. Sin embargo la impronta de Sergio sigue teniendo influencia en mi vida académica. Su forma de hacer docencia, la mejor que conocí, me dio las herramientas y la confianza necesarias para atacar problemas desconocidos con método y rigor matemático, escuela continuada brillantemente por Ernesto. La pulsión inicial que me llevó a pensar que podía resolver problemas de performance en redes de datos aplicando métodos de la teoría de control se la debo a Sergio. Luego, como suele ocurrir, la Tesis tomó rumbos diferentes. Sergio también nos hizo ver a muchos que se podía hacer ciencia desde la ingeniería, en la frontera entre la ciencia básica y la ciencia aplicada. Un concepto díficil de capturar por niños, iletrados, y también por funcionarios del sistema científico nacional.

También gracias a Jan Creelman, por recibirme cada año en su casa en Ottawa. Música clásica, galletas recién horneadas y arte batik: no había excusa para no concentrarse en la Tesis. 18 Melrose Av. es mi segundo hogar en Canadá.

También en Ottawa, gracias a los Dres. Jackie (Qi) Liu, Shafagh Jaffer y Mohammad Moallemi por hacer que mis visitas al laboratorio de Gabriel en Carleton University sean tan agradables como productivas científicamente, por medio de su hospitalidad y de largas discusiones científicas. Por los mismos motivos pero en el laboratorio de Ernesto en CIFASIS-CONICET, Rosario, a mis compañeros Gustavo Migoni, Fede Bergero, Mario Bortolotto y Fer Fontenla.

En Buenos Aires, a mis compañeros de la Oficina 10, Luciano Grippo, Martin Safe y Rosana Matuk, por las muchas charlas justo a tiempo antes de tirar la computadora por la ventana. Salud, doctores !

En particular, gracias a mis valientes tesistas de licenciatura Ivan Ramello (UNR) y Matias Bonaventura (UBA), por elegirme como director y por todo el empeño puesto en lograr resultados excelentes, a los cuales esta Tesis debe parte de sus aportes.

A mis amigos de la vida y de la facultad que siempre están, lejos o cerca en el tiempo y en la distancia: Guillito, Juan Pablo, Damián, Caprice, Juani, Darío, José y Polaco.

A mis viejos Adriana y Daniel, por apoyarme siempre en mis decisiones y haberme dado todo para que las pueda afrontar. No es poca cosa felicitar a un hijo por dejar un sueldo generoso y estable en una industria multinacional para optar por una beca breve y magra en una carrera impredecible y subvalorada como la científica. A mi hermano Marcelo, por recordarme siempre que en realidad lo único que impora es la música. Largos días y placenteras noches, sai Marcelo de Santa Fe .

A mis suegros Hector y Pili, por las innumerables manos brindadas para sacar preocupaciones del camino y poder concentrar esfuerzos en la Tesis.

Finalmente, el agradecimiento más importante es para Nati. Solo con su amor infinito pude y pudimos tomar tantas decisiones impensadas, sin las cuales esta Tesis no hubiera sido posible. Cuatro años intensos ! Nuestra pequeña familia y esta Tesis nacieron y crecieron a la par, junto con la pequeña Matilda, nuestra hermosa hija "fruto de la ciencia".

La confección del presente documento fue posible solo gracias a Pato. Infalible amigo de Pocoyó y primer palabra pronunciada por Matilda, supo pacificar a mi niña allí desde la ventana de youtube sobre el margen derecho de mi pantalla, mientras el editor de LaTeX acumulaba laboriosamente capítulos sobre el margen izquierdo. Gracias Pato.

Resumen

En la presente Tesis se desarrollan nuevas herramientas teóricas y prácticas para el modelado y simulación híbridos de redes de datos.

Se propone una metodología integradora basada en la especificación de sistemas de eventos discretos (DEVS) gracias a la cual es posible modelar y simular sistemas complejos eligiendo el modo de representación más conveniente para cada subsistema, lo que permite la coexistencia de modelos de tiempo discreto, de eventos discretos y/o continuos.

En particular se orienta el estudio al diseño de controladores de calidad de servicio de redes a distintos niveles de granularidad, promoviendo la aplicación de la Teoría de Control, y con miras a la implementación final en hardware a tiempo real.

Para poder modelar y simular este tipo de sistemas dentro del marco del formalismo DEVS se debió extender al mismo incorporando elementos estocásticos, y también se debieron extender los métodos de integración numérica existentes para considerar la presencia de retardos.

Por lo tanto, esta Tesis presenta una nueva herramienta teórica que extiende el formalismo DEVS brindándole la capacidad matemática formal de expresar procesos estocásticos, obteniendo así un nuevo formalismo denominado STochastic DEVS (STDEVS). Esto permite a DEVS representar de forma consistente los procesos aleatorios que dominan gran parte de los patrones de tráfico y de los algoritmos de control en redes de datos.

Otra nueva herramienta extiende la familia de métodos de integración numérica QSS que permiten a DEVS aproximar soluciones de sistemas continuos expresados por medio de ecuaciones diferenciales ordinarias (ODE). La nueva extensión denominada Delay QSS (DQSS) permite utilizar DEVS para aproximar soluciones de ecuaciones diferenciales con retardos (DDE), una clase de modelos fluídos que surgen en importantes aplicaciones de control de calidad de servicio en redes de datos.

Gracias a las nuevas herramientas antes mencionadas será posible modelar flujos híbridos de paquetes, combinando la ventaja de la eficiencia computacional de una simulación fluída con la ventaja del detalle granular provisto por una simulación discreta, bajo un formalismo matemático unificado.

Estas herramientas y técnicas trascienden el dominio de aplicación de las redes de datos, proveyendo soluciones generalizadas para la disciplina de modelado y simulación.

En esta Tesis se desarrollan también varias herramientas prácticas para facilitar la transición de modelos DEVS desde entornos de modelado, simulación y verificación hacia plataformas de hardware para su validación e implementación final. Se desarrolló un laboratorio virtual y portable para experimentar con un

hardware especializado en procesamiento de tráfico de red (Network Processor). El laboratorio incluye nuevas bibliotecas de comunicación entre un simulador DEVS y el hardware externo, permitiendo la ejecución embebida de modelos DEVS en tiempo real y en modalidad Hardware–In–The–Loop. Esto permite mantener la continuidad de los diseños de controladores desde su análisis inicial hasta su implementación como producto de software final, siguiendo el enfoque de la ingeniería basada en modelos.

Tanto la metodología propuesta en esta Tesis como los resultados originales obtenidos tienen como fin último ofrecer un marco teórico–práctico de modelado y simulación robusto y eficiente que facilite la ingeniería interdisciplinaria entre las comunidades de Teoría de Control, Redes de Datos e Ingeniería de Software.

Abstract

In this Thesis new theoretical and practical tools are introduced for the hybrid modeling and simulation of data networks. An integrative methodology based on the Discrete EVent Systems specification (DEVS) is proposed, making it possible to model and simulate complex systems by choosing the most convenient representation for each subsystem, allowing discrete time, discrete event and continuous models to coexist. In particular, the study is oriented towards designing quality of service controllers for networks at diverse granularity levels, enabling the adoption of Control Theory, and aiming at the final implementation of algorithms in real–time hardware.

In order to model and simulate this class of systems within the DEVS framework, it had to be extended by adding stochastic elements and extending the numerical integration methods to account for the presence of delays.

Thus, this Thesis introduces a new theoretical tool that extends the DEVS formalism equipping it with the ability to express stochastic systems in a mathematically formal and sound way. The newly obtained formalism is called STochastic DEVS (STDEVS), and allows DEVS to represent the stochastic phenomena dominating most of the traffic patterns and control algorithms in data networks.

Another new tool extends the QSS family of numerical integration methods which allow DEVS to approximate solutions of continuous systems expressed with ordinary differential equations (ODE). The new extension is called Delay QSS (DQSS) and enables DEVS to approximate solutions of delay differential equations (DDE), a class of continuous models that appear in distributed control applications for quality of service in data networks.

The new tools transcend the data networking application domain, constituting new generalized solutions for the modeling and simulation fields.

Many other practical tools are introduced in this Thesis to facilitate the transition of DEVS models from modeling, simulation and verification environments, to hardware platforms for validation and final implementation purposes. A new virtual laboratory was set up to experiment with real specialized traffic processing hardware. The lab includes new libraries and development environments that allow keeping the continuity of DEVS models, executing them as embedded standalone executives running in real–time.

Capítulo 0

Extended Summary

In this Thesis new theoretical and practical tools are introduced for the modelling and simulation (M&S) of hybrid systems with a focus on data networks.

An integrative methodology based on the Discrete EVent Systems specification (DEVS) is proposed, making it possible to model and simulate complex systems by choosing the most convenient representation paradigm for each subsystem, allowing discrete time, discrete event and continuous models to coexist. In particular, the study is oriented towards designing quality of service controllers for networks at diverse granularity levels, enabling the adoption of Control Theory, and aiming at the final implementation of algorithms in real–time hardware.

In order to model and simulate this class of systems within the DEVS framework it had to be extended by adding stochastic elements in a formal way to compensate the fact that the DEVS mathematical structure is essentially defined by means of deterministic functions. Also, existing numerical integration methods had to be extended to account explicitly for the presence of non-negligible delays (prevalent in data network dynamics) when fluid flow approximation models are used (i.e., differential equations).

Accordingly, this Thesis introduces a new theoretical tool that extends the DEVS formalism equipping it with the ability to express stochastic systems in a mathematically formal and sound way, resorting to the theory of probability spaces. The newly obtained formalism is called STochastic DEVS (STDEVS), and allows DEVS to represent stochastic phenomena such as those dominating most of the traffic patterns and control algorithms used in data networks.

Another novel theoretical tool extends the Quantized State Systems (QSS) family of numerical integration methods which allow DEVS to approximate solutions of continuous systems expressed with ordinary differential equations (ODE) of up to third order accuracy. The new extension is called Delay QSS (DQSS) and enables DEVS to approximate solutions of delay differential equations (DDEs), a class of continuous models that appear in distributed control applications for quality of service in data networks (and also in several modeling applications in natural and social sciences).

These new tools transcend the data networking application domain, constituting new generalized solutions for the M&S fields in a broad sense.

Many other original tools of practical nature are introduced in this Thesis to facilitate the transition of DEVS models of network systems from mode-

ling, simulation and verification environments, to target hardware platforms for validation and final embedded implementation purposes.

A novel, reusable virtual laboratory was set up to experiment with specialized traffic processing hardware. The virtual lab includes a novel general-purpose Eclipse-based DEVS graphical development environment, along with new low level specific-purpose networking libraries. It allows keeping the continuity of DEVS models originally simulated in a desktop computer by automatically porting and executing them as standalone executives embedded into target hardware platforms in real–time.

0.1. Motivation and Application Domain

In the last two decades packet communication networks have become a central component in almost every branch of engineering. Either as a constitutive part of technological solutions or as services and subject of study on their own, data networks have experienced an exponential growth in both scale of adoption and technological complexity. As a consequence, to a good extent network engineering has been growing ahead of science. Currently there is a lack of theoretical and practical tools that can fully explain several of the phenomena affecting the quality of service of networking infrastructures, and consequently, of the end–user level applications that rely on them.

In this context, a discipline has gained an increasing interest: the automatic control of the quality of service of data networks, which deals with the design of optimal control strategies to assign efficiently finite network resources to the processing of packet flows competing for them.

Usually the demand imposed by packet flows is unpredictable, raising the challenge of designing flexible networks. They should accommodate an acceptable average traffic load along with sporadic peaks of bandwidth consumption, at a viable cost, without incurring in oversized designs, and guaranteeing quality requisites in terms of maximum delays, uptime, effective bandwidth, etc.

As networks grew in size and complexity it became evident the need to abstract and analyze their behavior resorting to models and computer simulation to help designing efficient controllers. However, there are certain characteristics of networks that present important theoretical and practical challenges to the known M&S methods.

Given the existing diversity of networks and applications using them, the concept of quality of service can change dramatically, and with it the associated control objectives and techniques to be adopted. The community devoted to the M&S of networks oscillates from the development of generalized models that help describing most problems with a high level of abstraction to the development of highly specialized models that focus on solving specific issues in bounded problem domains. Accordingly, various and disparate modeling paradigms are adopted, more or less distant from the underlying simulation algorithms selected to execute them.

Therefore, it is natural to face integration challenges that hamper the combination of research results across different approaches. This hinders progress in the field of (M&S)–based network research.

The objectives of this Thesis focus on providing an integrative formal framework for modeling, simulation and implementation of data networks controllers,

which will be suitable for a vast range of solution approaches and types of models, under a single formalism, fostering cooperative research.

0.2. Different types of network models

Models of networks and their controllers cover problem domains at different hierarchical levels, involving dissimilar temporal dynamics and heterogeneous specification languages.

In upper control hierarchies are global policy controllers for traffic prioritization, assigned to different types of data flows, involving control actions a few times in a minute, hour or even day.

At a lower, intermediate level, the traffic shaping controllers tailor the allocation strategies of available resources trying to enforce the global policies set, resorting to dynamics in the order of seconds.

At the lowest level, control algorithms take real-time decisions at the granularity of individual packets (dynamics of milliseconds or microseconds).

This scenario makes it difficult to integrally design and test performance control systems and quality of service policies.

In particular, the task becomes complex to verify and validate the effects of changes introduced in a specific level of algorithms into the performance of the integral system.

Furthermore, digital communication networks present difficulties for M&S when their size grows. More detailed representations (at the *individual packet* level) consist of discrete event models able to capture very accurate behaviors. These models present the disadvantage of having a very high computational cost for simulating complex network topologies and/or considerable traffic intensities. Besides, they are not useful for the analysis of global network properties.

Alternatively, less detailed representations (at the *packet flow* level) consist of continuous models that approximate network dynamics using averaged values of variables. These continuous models allow to simulate efficiently (in very low times) systems that exhibit high complexity in their original discrete formulation. Their mathematical representation in the form of differential equations allows also for analytical treatment, which among other advantages, makes it possible to design network controllers based on Control Theory. Thus, very powerful analytical techniques can be applied to study the stability, robustness and performance of the obtained designs. In return, the continuous approximations have the limitation of capturing only slow or steady state dynamics, since all variables are described using time averages.

Other relevant dimensions worth considering are a) the stochastic characteristic of network traffic (and also topology), and b) the need to implement the controllers designed to operate in real–time and in specific–purpose hardware. It is fundamental to ensure a solid formal representation of random processes governing demand patterns and other stochastic phenomena in networks, and to keep such representations consistent throughout all design stages. At the final implementation stage, it is also very relevant to retain the validity of the models obtained across all previous stages, being able to reuse them directly as the actual final pieces of software running in real–time, interacting with specialized hardware for network control.

0.3. Proposed integrative solution for M&S of data networks.

This Thesis proposes an alternative integrative framework for applying hybrid techniques of M&S under a unifying formalism, avoiding the need for change of formalisms (or tools) to study the systems described. Specifically, we resort to DEVS (*Discrete EVent Systems Specification*) as the formal and practical integrative framework for M&S of data networks and control algorithms.

DEVS can exactly represent any discrete time and discrete event system, and approximate continuous systems as accurately as desired. DEVS allows to model exactly any network protocol, and by means of techniques for approximating continuous systems, to describe also continuous approximations of network flows, thus making discrete and continuous models coexist without requiring a change of formalism. Moreover, there exist tools for transforming DEVS models into pieces of software running in real–time avoiding to tailor (or even rewrite) the logic of the models.

In this thesis, we will analyze the feasibility of applying DEVS as an integrative solution. In doing so we will identify limitations of theoretical and practical nature, and will propose new tools to overcome them. Some of these tools transcend the application domain of data networks, providing generalized solutions for the M&S discipline un general.

0.4. Summary of chapters

0.4.1. Summary of Chapter 2: *Preliminary Concepts*

The chapter provides preliminary technical concepts that serve as a starting point to highlight the under–addressed needs identified, and the solutions contributed. It begins with an overview of *hybrid dynamical systems* in Section 2.1 to characterize the main representations which can be used in modeling data networks: discrete-time systems, discrete event systems and continuous systems. It highlights the problem of the interaction of discrete signals with continuous signals, producing discontinuities in the classical numerical integration methods. It is mentioned here for the first time QSS (*Quantized State Systems*) that replaces time discretization by the quantization of state variables, and DDE (*Delay Differential Equations*) in which the dynamics of a system depend on both its current state and also its dynamic history.

The QSS and DDE will have a central role in the development of new tools for developing continuous models for networks. In this context, DEVS is proposed as the M&S integrative formalism for hybrid dynamical systems in general.

Immediately Sections 2.2 and 2.3 present in detail DEVS and QSS, highlighting their most salient characteristics and the relationship between them.

Afterwards, Section 2.4 presents two software tools based on DEVS M&S of hybrid systems. The first is PowerDEVS, a tool to graphically model and simulate engineering systems in a block-oriented fashion. The second is ECD++ (*Embedded CD++ Simulator*), a specialized version of the general purpose CD++ for real–time embedded M&S purposes. These software tools will support the implementation, verification and validation of the contributions made in this Thesis. To conclude, in Section 2.5 DEVS is put in context with the modeling

of data networks, and through a motivating scenario study (see Figure 2.11) unmet needs are identified that arise when trying to apply DEVS as an integrative formalism. In that study scenario, three collaborative network controllers are proposed and explained: supervisory, congestion and admission, operating over a typical pattern of nodes running the TCP (*Transport Control Protocol*) protocol and served by a router node that enforces AQM (*Active Queue Management*).

0.4.2. Summary of Chapter 3: *STDEVS - A Novel Theoretical Framework for STochastic DEVS specification.*

This chapter is devoted to the analysis and solution of the first identified need; providing the DEVS formalism with the ability to express stochastic phenomena in a mathematically consistent and correct way.

Sections 3.1 and 3.2 raise the roots of the problem and review a history of the rather scarce previous attempts to relate DEVS with random processes, finally putting out the need to resort to the theory of probability spaces and measurable functions. In Section 3.3 begins with the intuitive idea of introducing random number generators as part of the DEVS functions, and loops through formal inconsistencies produced by this naive approach.

Then, Section 3.4 presents the solution proposed consisting of a new formalism STDEVS (*STochastic Discrete EVent Systems Specification*) (see Eq. 3.1 and Figures 3.2 and 3.3), and develops a rigorous mathematical definition for atomic and coupled models. The main properties of STDEVS are discussed in Section 3.5, showing that it preserves the closure under coupling and providing a redefinition of the legitimacy condition for that property in a stochastic context. Shown later in Section 3.6 is that it is possible to build STDEVS models using deterministic DEVS models equipped with RND standard functions in their transition functions and that DEVS models with measurable functions are special cases of STDEVS, enabling to formally couple DEVS with STDEVS models. The chapter continues with Section 3.7 developing two case studies to illustrate the practical use of STDEVS: a model of a network load balancer, and a model of a networked automatic control system.

Main Results

We have presented a novel formalism for describing stochastic discrete event systems. Based on the system theoretical approach of DEVS and making use of the probability spaces theory, STDEVS (*STochastic Discrete EVent Systems Specification*) provides a formal framework for M&S of generalized nondeterministic discrete event systems. STDEVS is a general and comprehensive formalism that provides the ability to represent stochastic behavior over measurable infinite–dimensional spaces, along with the ability to represent traditional stochastic discrete event formalisms used widely for many applications (e.g., Markov chains, queueing networks, stochastic petri nets, etc.). Owing to its DEVS-based roots, STDEVS is also a system theoretic–based representation, which provides unique multi–modeling advantages for the specification of hybrid systems, and important simulation performance enhancements when dealing with heavily discontinuous hybrid systems.

Contributed publications

- An early formulation of the STDEVS idea was published in a local conference in [KC06].
- The final form of the formalism was presented at an international congress in [CKW08].
- Finally, a comprehensive and mathematically exhaustive journal manuscript was published [CKW10]. **This manuscript contains all the main technical contributions presented in this chapter.**

0.4.3. Summary of Chapter 4: *DQSS - A Novel DEVS-based Numerical Integration Method for Solving Delay Differential Equations.*

This chapter is devoted to the treatment and solution of another under–addressed identified need: that of simulating numerically DDE (*Delay Differential Equations*) with efficient handling of discontinuities. We start from the well–established previous results for state quantization-based numerical integration of ODE (*Ordinary Differential Equations*); more specifically, the QSS family of integration methods. The chapter introduces DQSS (*Delay Quantized State Systems*), a novel class of numerical integration algorithms for solving DDEs based on discrete event–oriented variable quantization rather than the classical time discretization approach. DQSS permits modeling adequately with DEVS continuous approximations of those dynamics of packet networks whose state variables are subject to non–negligible time delays (and therefore are affected by their history).

Section 4.1 (together with 2.3.1) presents a basic introduction to the QSS methods and an intuitive idea for their application to the solution of DDEs. In Section 4.2 the generalization of the concept of QSS applied to DDE is discussed and the resulting DQSS algorithms are presented in detail. Section 4.3, together with Appendix A.1 and A.3, discuss the numerical properties of the DQSS algorithms, specifically, their stability and convergence, including the corresponding theorems and proofs. Then, Section 4.4 presents simulations applying DQSS in four comparative performance studies using test models taken from the open literature. To conclude, in Section 4.5 detailed conclusions are offered about the implications of the contribution made, which transcends the application domain of packet networks representing a generalized advance for the discipline of M&S.

Main Results

We introduced a new class of numerical delay-differential equation (DDE) solvers. See Figures 4.3, 4.4 and 4.6. Numerical DDE solvers that are based on state quantization instead of time-slicing offer some striking and unparalleled properties that make these algorithms particularly well suited for the task at hand.

First, Quantized State System (QSS) methods are naturally asynchronous. These are variable-step methods, whereby each state variable is updated at its own pace. State variables with larger gradients are updated more frequently than state variables with little ongoing activity. For this reason, the QSS–based

solvers exploit implicitly and intrinsically the sparsity of the model to be simulated.

Second, we were able to show that an asymptotically stable linear time–invariant DDE system with constant delays leads to a QSS approximation that remains numerically stable for any quantum size. The granularity of the quantization influences the accuracy of the simulation results, but not the numerical stability.

Third, we were able to show that an input-to-state-stable non-linear time-varying DDE system with variable and even state-dependent delays leads to a QSS approximation that remains numerically stable for any sufficiently small quantum size. For any such system, there exists a maximum quantum size larger zero, such that its QSS approximation remains numerically stable for all smaller or equal quanta.

Fourth, we showed that as the quantum approaches zero in such a system, the numerical simulation trajectories converge to the analytical trajectories of the original DDE system.

Fifth, whereas the step-size control algorithms of time-slicing based solvers usually control the local integration error only, state-quantization based solvers control the global integration error. Hence state-quantization based solvers are more robust than their time-slicing based competitors.

Sixth, it was demonstrated by means of four different benchmark problems from the open literature that state-quantization based DDE solvers are frequently significantly more efficient than their time-slicing based competitors. The reason for their higher efficiency is partly because they naturally exploit the sparsity in the models to be simulated, and partly because the algorithms used for interpolating the delayed states are simpler and can thus be implemented more efficiently.

Seventh, the DQSS codes as implemented in PowerDEVS, a M&S environment with a graphical user interface similar to that of Simulink, are much easier to use than the dde23 (Matlab) and dde_solver (Fortran) codes that were previously available.

DQSS has no antecedents in the literature, constituting the first numerical integration algorithm of DDE based on the quantization of state variables.

Contributed publications

- A comprehensive and mathematically exhaustive journal manuscript was published [CKC10a]. **This manuscript contains all the main technical contributions presented in this chapter.**
- A poster has been also presented at a regional Winter School of Science in Informatics [CKC10b].

0.4.4. Summary of Chapter 5: *Model-Based Real-Time Embedded Simulation for Controlling Data networks.*

In this chapter we make contributions in the field of real-time embedded implementation of DEVS models.

We developed software tools and set up a hardware infrastructure to obtain a fully DEVS–based process that helps embedding models into special–purpose hardware targets and run the models in real–time while they interact with external hardware units.

The strategy maintains DEVS model continuity by encapsulating away the functionalities that provide interaction with external hardware. Special *Software/Hardware Mapper* atomic models provide the ability to interact with target–specific control interfaces.

Section 5.1 offers an overview of antecedents in formalisms and tools meant to facilitate the transition of system-level models towards their final real–time implementation into hardware.

In Section 5.2 we analyze a particular hardware relevant for this Thesis: an Intel Network Processor IXP2400 is adopted as a test bed platform for executing and validating real–time DEVS models in the context of a traffic control application. See Figure 5.3.

Section 5.3 introduces our novel adaptation of the ECD++ (*Embedded CD++ Simulator*) tool to make it able to run embedded into the IXP2400 network hardware. New ECD++ libraries are provided to communicate DEVS models with low level, IXP2400-specific packet handling libraries. See Figure 5.4.

Afterwards, in Section 5.4 we introduce a series of advanced tools for assisting the process of designing embedded systems for controlling network traffic. Firstly we present an advanced Graphical User interface for M&S with ECD++ based on the Eclipse development environment (see Figure 5.7). Secondly, we set up a virtualized and portable experimentation laboratory for developing DEVS models together with a high capacity network card based on the IXP2400 tool (see Figure 5.8). The section concludes with a practical example showing the step by step development of a supervisory control for traffic quality of service using the contributed set of tools, including a validation phase operating under real networking load.

Main Results

We obtained a set of new interface libraries that flexibly communicate input ports and output ports of ECD++ with specific hardware addresses of the Network Processor Intel IXP2400. We encapsulated the low level communication features into dedicated *DEVS Mapper* atomic models which can declare special *IXA Communication* variables. A postprocessor algorithm parses DEVS models and automatically generates the code of a stack of intermediate libraries that make those special IXA variables have their counterparts into the network processor's memory spaces. This guarantees model continuity of DEVS-based network controllers by separating platform independent model behaviour from platform dependent software/hardware interfaces.

We also developed the advanced CD++Builder version 2.0 to provide a more powerful Integrated Development Environment (IDE). It includes all M&S tasks (modeling, compiling, simulation execution, and analysis), supplies editors that support the complete modeling cycle to be performed graphically, includes C++ code templates to aid in the implementation of atomic models (while keeping the graphical representation of these models consistent with their C++ code) and supports extensibility through development of new features into the IDE (including automated regression testing capabilities). CD++Builder2.0 was built

upon Eclipse's extensible mechanisms, which allows adding new plugins independently from the rest of the features. Graphical editors guarantee easy-to-use interfaces and simplify extensibility. CD++Builder supports complete graphical modeling, combining CD++ coupled models and DEVS-Graphs, allowing non expert developers to model real-world systems utilizing the DEVS formalism. The CDT Eclipse plug-in is used for developing more advanced models using C++ (whenever this is needed), allowing a combination of graphical and non-graphical models, and making the coding of C++ models easier. We improved issues about usability and modeling limitations using advanced editors, a tool for easier model reuse, a coupled model editor with automatic discovery, and new install and update mechanisms.

An example of this is the application of CD++Builder2.0 in our new Virtual Laboratory for Model-Based Development for Network Processors (NP) and Embedded-CD++. We benefited from CD++Builder2.0 extensibility capabilities and built plugins that deal transparently with intricacies imposed by the target hardware (specific procedures for compiling, downloading and monitoring models for their real–time execution on the target platform). It also provides an integrated environment for mixing DEVS models with low-level hardware-specific drivers, making the simulator interact with real network signals in real–time.

The Virtual Laboratory (see Figure 5.8) is a portable setup which includes all the tools and configurations required to start experimenting out–of–the–box with ECD++ and IXP2400, featuring an NP-based high capacity routing board RadiSys ENP2611, a router, a traffic generator, etc.

The practical test cases conducted with real network traffic showed that final implementation and validation of DEVS-based network controllers can be carried out successfully, completely eliminating the need for adapting neither logic nor structure when evolving from standalone simulations (verification phase) to Hardware-In-The-Loop executions (validation phase). See e.g. Figures 5.9, 5.10 and 5.11. The methodology promotes engineering solutions fully based on DEVS M&S.

Contributed publications

- A first version of the advanced graphical user interface (CD++Builder 2.0) was presented at an international congress [BWC10a] (*Conference Runner-Up Best Paper Award*)
- Also a poster was presented at a regional Winter School of Science in Informatics [BWC10b].
- An extended version of the previous article was then published as a journal manuscript in [BWC12].
- A comprehensive and detailed description of the end-to-end process for implementing a network controller using the new Virtual Laboratory was published in [CRBW12] at an international congress.
- **The next two contributions are based partially on results obtained in this chapter:**

- The design of the supervisory network control embedded into the IXP2400 processor was first suggested in [JWBC09] at an international congress.
- The article [WC11] was published in a M&S practitioner's magazine based partially on the model continuity concepts developed in this chapter in the context of embedded systems.

0.4.5. Summary of Chapter 6: *Application: M&S-based application of Control Theory to a Packet Admission Control Problem.*

In this first application–oriented chapter we present an integrative M&S methodology based on DEVS to design and implement a Quality of Service control system applying classical Control Theory. Section 6.1 overviews several real–time systems that apply classical Control Theory to computing and/or networking systems, which are considered themselves the objects to be controlled.

Section 6.2 describes the most salient antecedents of Admission Control systems (AC) while Section 6.3 presents a case study applying Control Theory to obtain a Proportional–Integral (PI) class of controller for an Admission Control problem. Afterwards, in Section 6.4 we propose our integrative M&S methodology based on DEVS to assist the process of analysis, design, verification, implementation and validation of the PI controller studied in the previous section. The problem at hand is of a strong hybrid nature, and requires a formal treatment to guarantee the applicability of the continuous PI controller for discrete event systems (as it is the case of a packet network system).

Main Results

We showed how to adopt the DEVS M&S framework as the basis of an end–to–end methodology for applying Control Theory into a network admission control problem, which yields an stochastic hybrid system. We showed how the usual challenges that arise in hybrid systems due to the needs for switching between M&S paradigms disappear by following a full DEVS–based methodology. Thanks to the QSS numerical methods, continuous systems can be seamlessly integrated with discrete-event systems, providing a very suitable common framework to integrate heterogeneous disciplines in the area of control of real-time computing and networking systems. We first analyze and design a fully continuous approximation of the network and the controller (using differential equations, see Figure 6.2). Then we discretize in time the equations of the controller, keeping the network model in its fluid approximation (this step envisions the final implementation of the controller in a digital platform, i.e., a temporally discrete, clock–driven environment) See Figure 6.4. A verification due to the discretization step is performed in this phase. Finally, as a last verification step before the validation in a real setup, the fluid approximation of the network is replaced by an exact discrete event network model, much closer to the real system (see Figure 6.6). Further reference is made to methods for translating semiautomatically the discrete–time DEVS model of the PI controller from its PowerDEVS implementation to an ECD++ implementation which is naturally portable to run embedded into the IXP2400 network processor, as it was extensively described in the previous chapter.

Contributed publications

- The DEVS–based end-to-end methodology for hybrid control of embedded networking systems was presented at an international congress in [CKW09].

0.4.6. Summary of Chapter 7: *Application: M&S-based study of a Network Congestion Control Problem (TCP/IP).*

In this second application–oriented chapter we apply the new DQSS methods to develop a fluid representation of a congestion control algorithm of type TCP/AQM (*Distributed Congestion Control*). In Section 7.1 we specify in detail the expected behaviour for the existing *sliding window* flow control algorithm AIMD (*Additive Increase Multiplicative Decrease*) and the RED (*Random Early Detection*) mechanism, which operate at the extreme nodes and also at the intermediate routing element that connects them, respectively. See e.g. Figures 7.1 and 7.2.

In Section 7.3 we develop in PowerDEVS a DEVS–based fluid approximation of TCPAQM allowing to simulate in very convenient times the main dynamical characteristics of the system. See Figure 7.4.

Afterwards, also in the PowerDEVS tool, we developed several discrete event models to simulate exactly and in more detail the behaviour of individual network packets flowing under the control of the TCPAQM algorithm. This time we modeled the exact algorithmic and physical properties of network protocols and physical media, respectively. See Figure 7.7.The results obtained in this step allow to verify the level accuracy of the fluid approximation developed in the previous step, as well as to study in detail discrete properties closer to the real system represented.

Then, in Section 7.4 we developed a novel modeling strategy of so-called *hybrid flows* for combining simultaneously the computational efficiency of a fluid flow approximation with the advantages of the granular details provided by a discrete simulation. See Figures 7.12, 7.13, and 7.15.

Main Results

By means of DEVS we obtained a hybrid representation (both continuous and discrete, concurrently) of flows of entities (network packets in this case) under a unified mathematical formalism, and under the same tool. It became evident the benefit of retaining packet–wise detailed information (e.g., for a discrete witness traffic flow) along with the reduction of simulation times for e.g. a fluid background traffic flow. This contrasts with the usual approach of having the whole traffic (the witness flow and the background flows) simulated in a fully discrete event fashion when detailed per packet information is required.

While developing the test case systems for this and the previous application–oriented chapters, twelve novel general–purpose network–oriented DEVS atomic models were created, which constitute a new reusable network library for PowerDEVS. It is depicted in detail in Appendix B.1. These models, in turn, make use of new auxiliary (lower level) libraries that a) provide new data structures for

representing network packets with hierarchical composition of protocol structures following the Open Systems Interconnection (OSI) model and b) provide easy access from models in PowerDEVS to generate flows of stochastic events according to parameterizable probability distributions (using the high quality random number generators provided by the open source GSL library (GNU Scientific Library)).

Contributed publications

- These original results remain unpublished.

  The strategy of the simultaneous hybrid representation of flows and the very detailed DEVS–based implementation of the TCP congestion control mechanism remain original to the knowledge of the author. Currently, an undergraduate thesis is being developed based on these results.

Índice general

Índice de Acrónimos

Índice de figuras

Capítulo 1

Introducción

En las últimas dos décadas las redes de comunicación de datos orientadas a paquetes se han transformando en un componente cada vez más importante en casi todas las ramas de las ingenierías. Ya sea como parte constitutiva de soluciones tecnológicas o como servicio y objeto de estudio en sí mismas, su crecimiento ha sido exponencial tanto en adopción masiva como en complejidad tecnológica, y en gran medida la ingeniería se adelantó a la ciencia. Actualmente no se cuenta con herramientas teóricas ni prácticas para poder explicar, predecir y caracterizar acabadamente muchos de los fenómenos que afectan a la calidad de servicio de las redes, y en consecuencia, a la calidad de las aplicaciones tecnológicas que las utilizan.

En este contexto, una disciplina de gran interés es el control de calidad de servicio de redes, que diseña estrategias de control óptimas y eficientes para asignar recursos finitos de infraestructura de red al procesamiento de flujos de paquetes que compiten por acceder a la misma.

Generalmente la demanda impuesta por los flujos es impredecible, presentando el desafío de diseñar una red flexible, que permita un consumo promedio aceptable pero con ciertos picos esporádicos, a un costo viable y sin incurrir en el sobredimensionamiento, a la vez que garantice requisitos de calidad respecto de retardos máximos, tasa de disponibilidad, ancho de banda efectivo, etc.

A medida que las redes crecieron en tamaño y complejidad se hizo evidente la necesidad de analizar su comportamiento y diseñar sus controles basándose en modelos y en simulación por computadora. Sin embargo, existen características de las redes de datos que presentan desafíos teóricos y prácticos importantes a los métodos conocidos de modelado y simulación.

Por la diversidad de las redes y las aplicaciones que las utilizan, el concepto de calidad de servicio puede cambiar notablemente, y con ello los objetivos y técnicas de control. La comunidad dedicada al modelado y simulación de redes oscila entre el desarrollo de modelos generalistas que permitan describir la mayor cantidad de problemas con un alto nivel de abstracción y modelos especializados que se enfocan en resolver eficientemente dominios acotados de problemas. Se utilizan variados paradigmas de modelado, mas o menos alejados de los mecanismos de simulación que los implementan, también diversos.

Surgen entonces problemas evidentes de integración, haciendo difícil la combinación de resultados de investigación entre distintas especialidades.

Distintos tipos de modelos de redes

Los modelos de redes y sus controles abarcan dominios de problema a distintos niveles jerárquicos, involucrando dinámicas temporales disímiles y lenguajes de especificación heterogéneos. En las jerarquías de control superiores se encuentran las políticas globales de priorización de tráfico asignadas a distintos tipos de flujos de datos, implicando acciones de control unas pocas veces por minuto, hora o día. A un nivel intermedio, los algoritmos de perfilado de tráfico modifican las estrategias de asignación de los recursos disponibles intentando hacer cumplir las políticas globales, con una dinámica del orden de los segundos. En el nivel más bajo de todos, los algoritmos de actuación toman decisiones en tiempo real con granularidad a nivel de paquetes individuales. Este escenario hace difícil el diseño y test integral de sistemas de control de desempeño de red y sus políticas de calidad de servicio. En especial se torna compleja la tarea de verificar y validar los efectos producidos por la introducción de cambios en algoritmos de un nivel específico sobre el comportamiento del sistema integral.

Por otro lado, las redes digitales de comunicaciones presentan dificultades para su modelado y simulación cuando su tamaño crece. Las representaciones más detalladas (a nivel de paquetes individuales) consisten en modelos de eventos discretos capaces de capturar comportamientos exactos. Estos modelos presentan la desventaja de tener un costo computacional muy elevado para simular topologías de red complejas y/o intensidades de tráfico considerables. Tampoco son útiles para realizar análisis de propiedades globales.

Las representaciones menos detalladas (a nivel de flujos de paquetes) consisten en modelos continuos que aproximan la dinámica de la red mediante variables promediadas.

Estos modelos continuos permiten simular en tiempos muy bajos sistemas que en su formulación original discreta tienen muy alta complejidad. La representación matemática como ecuaciones diferenciales, a su vez, permite el tratamiento analítico que, entre otras cosas, hace posible diseñar controladores de redes basados en la Teoría de Control. De esta manera pueden aplicarse técnicas analíticas muy potentes para estudiar la estabilidad, robustez y desempeño de los diseños obtenidos.

Como contrapartida, las aproximaciones continuas presentan la desventaja de capturar sólo dinámicas lentas o de régimen estacionario, ya que las variables se describen mediante promedios temporales.

Otras dimensiones relevantes a considerar son la característica estocástica de las redes, y la necesidad de implementar los controladores diseñados operando en tiempo real. Es de fundamental importancia garantizar una representación formal sólida de los procesos aleatorios que rigen patrones de demanda y otros fenómenos estocásticos en las redes, y permitir que dicha representación se conserve de manera coherente hasta la etapa final de implementación. En dicha etapa, además, es muy importante conservar la validez de los modelos de las etapas previas, y tener la capacidad de utilizarlos como piezas de software operando en tiempo real e interactuando con el hardware para control de red.

Solución integradora para modelado y simulación de redes

En esta Tesis se propone una alternativa integradora consistente en aplicar técnicas híbridas de modelado y simulación bajo un formalismo unificador, evi-

tando la necesidad de cambiar de formalismo o herramienta para estudiar los sistemas descritos. Concretamente, se recurre al formalismo para Especificación de Sistemas de Eventos Discretos (DEVS, por *Discrete EVent Systems Specification*) como marco integrador formal y práctico para el modelado y simulación de redes de datos y sus algoritmos de control.

DEVS puede representar exactamente cualquier sistema de tiempo discreto y de eventos discretos, y puede aproximar sistemas continuos con tanta precisión como se desee. Especificación de Sistemas de Eventos Discretos (DEVS) permite modelar de modo exacto cualquier tipo de protocolo de comunicaciones, y por medio de extensiones para representación de sistemas continuos, permite describir aproximaciones continuas de las redes, haciendo coexistir los modelos discretos y continuos sin requerir un cambio de formalismo. Aún más, existen herramientas que permiten transformar los modelos DEVS originales en piezas de software ejecutando en tiempo real evitando la reescritura de la lógica de los modelos.

En la presente Tesis, se analizará la viabilidad de aplicar DEVS como solución integradora, se encontrarán limitaciones de orden teórico y práctico, y se propondrán nuevas herramientas para intentar superarlas. Parte de estas herramientas trascenderán el dominio de aplicación de las redes de datos, proveyendo soluciones generalizadas para la disciplina de modelado y simulación.

1.1. Organización de la Tesis

El presente primer capítulo introductorio presenta una descripción de los temas centrales de la Tesis, la problemática abordada y las soluciones propuestas, describiendo los resultados originales obtenidos y las publicaciones científicas elaboradas.

En el Capítulo 2 se proveen los conceptos técnicos preliminares que servirán como punto de partida para poner en evidencia las necesidades identificadas y las soluciones aportadas. Se comienza con una descripción general de *Sistemas Dinámicos Híbridos* en la Sección 2.1 para caracterizar las principales representaciones a las que puede recurrirse en el modelado de redes de datos: sistemas de tiempo discreto, sistemas de eventos discretos y sistemas continuos. Se pone de manifiesto la problemática de la interacción de las variables discretas y continuas, produciendo discontinuidades en los métodos clásicos de integración numérica. Se menciona aquí por primera vez los Sistemas de Estados Cuantificados (QSS, por *Quantized State Systems*) que reemplazan la discretización temporal por la cuantificación de las variables de estado, y las Ecuaciones Diferenciales con Retardo (DDE, por *Delay Differential Equations*) en las cuales la dinámica de un sistema depende tanto de su estado actual como de su historia. Los Sistemas de Estados Cuantificados (QSS) y las Ecuaciones Diferenciales con Retardo (DDE) tendrán un rol central en el desarrollo de nuevas herramientas para el modelado continuo de redes. En este contexto, se sugiere a DEVS como formalismo de modelado y simulación integrador para sistemas dinámicos híbridos en general. Inmediatamente la Secciones 2.2 y 2.3 presentan en detalle el formalismo DEVS y los QSS, respectivamente, proveyendo sus características más sobresalientes y la relación que existe entre ambos. Se continúa con la Sección 2.4 que presenta dos herramientas de software basadas en DEVS para modelado y simulación de sistemas híbridos. La primera de ellas es PowerDEVS,

una herramienta que permite modelar gráficamente sistemas ingenieriles y simularlos eficientemente. La segunda es el Simulador CD++ Embebido (ECD++, por *Embedded CD++ Simulator*), una versión especializada para ejecución embebida en tiempo real del simulador de propósito genérico CD++. Estas herramientas de software servirán de soporte para la implementación, verificación y validación de los aportes realizados a lo largo de la Tesis. Concluyendo con los conceptos preliminares, en la Sección 2.5 se pone en contexto a DEVS con el modelado de redes de datos, y por medio de un escenario de estudio motivador se identifican necesidades no cubiertas que surgen al pretender aplicar DEVS como formalismo integrador. En el escenario de estudio se plantean 3 controles de red autónomos y colaborativos: de supervisión, de congestión y de admisión, operando en un esquema típico de nodos con protocolo Protocolo de Control de Transporte (TCP) y un nodo enrutador con administración activa de colas.

En los Capítulos 3,4 y 5 se presentan los principales aportes teóricos y prácticos para posibilitar la obtención de una metodología integradora completamente basada en DEVS. Luego, en los Capítulos 6,7 se ponen en práctica las nuevas herramientas y métodos obtenidos, aplicándolos al escenario de motivación planteado en el Capítulo 2. La Tesis finaliza con el Capítulo 8 con una enumeración de problemas abiertos y las conclusiones generales del trabajo realizado.

Herramientas teóricas y prácticas

El Capítulo 3 se aboca al análisis y solución de la primer necesidad identificada; se trata de proveer al formalismo DEVS con la capacidad de expresar fenómenos estocásticos en forma consistente y matemáticamente correcta. Las Secciones 3.1 y 3.2 plantean el problema y repasan antecedentes de intentos previos para relacionar DEVS con procesos aleatorios, poniendo finalmente de manifiesto la necesidad de recurrir a la teoría de Espacios de Probabilidad y funciones medibles. En la Sección 3.3 se trata cada uno de los tópicos teóricos de DEVS que justifican la necesidad de reformular el formalismo. Se parte de la idea intuitiva de introducir generadores de números aleatorios como parte de las funciones DEVS, y se recorren las inconsistencias formales que esto genera. Luego, en la Sección 3.4 se presenta la solución consistente en un nuevo formalismo para la especificación de Sistemas de Eventos Discretos Estocásticos (STDEVS, por *STochastic Discrete EVent Systems Specification*), desarrollando la definición matemática rigurosa para modelos atómicos y acoplados. Las propiedades principales de especificación de Sistemas de Eventos Discretos Estocásticos (STDEVS) se analizan en la Sección 3.5, mostrando que preserva la clausura bajo acoplamiento y proveyendo una redefinición para la propiedad de legitimidad en un contexto estocástico. Se muestra luego en la Sección 3.6 que es posible construir modelos STDEVS utilizando modelos DEVS deterministas equipados con funciones RND estándar en sus funciones de transición y que los modelos DEVS con funciones medibles son casos particulares de STDEVS, habilitando formalmente a acoplar modelos DEVS y STDEVS. El capítulo continúa con la Sección 3.7 desarrollando dos casos de estudio para ilustrar el uso práctico de STDEVS: un modelo de un sistema balanceador de carga de red, y un modelo de una aplicación de control automático interconectado en red. En la Sección 3.8 se cierra el capítulo con conclusiones sobre el aporte realizado.

El Capítulo 4 por su parte, se dedica al tratamiento y solución de otra de las necesidades identificadas: la de poder simular numéricamente Ecuacio-

nes Diferenciales con Retardo (DDE), partiendo de resultados previos para la integración por cuantificación de estados de Ecuaciones Diferenciales Ordinarias (ODE); más específicamente, partiendo de los métodos QSS.

Este capítulo introduce Sistemas de Estados Cuantificados con Retardo (DQSS, por *Delay Quantized State Systems*), una nueva clase de algoritmos de integración para resolver numéricamente DDE basados en técnicas de cuantificación de estados en lugar de la técnica clásica de discretización del tiempo. Sistemas de Estados Cuantificados con Retardo (DQSS) permitirá modelar adecuadamente con DEVS aproximaciones continuas de aquellas dinámicas de las redes de datos en que variables de estado son afectadas por sus valores en el pasado. La Secciones 4.1 y 2.3.1 presenta la idea básica de los métodos QSS y una idea intuitiva de su aplicación a la resolución de DDE. En la Sección 4.2 se discutirá la generalización del concepto de QSS aplicado a DDE y se explicarán los algoritmos DQSS en detalle. La Sección 4.3, junto con los Apéndices A.1 y A.3, discutirán las propiedades numéricas de los algoritmos DQSS, específicamente, su estabilidad y convergencia. Finalmente la Sección 4.4 presenta simulaciones DQSS en cuatro estudios comparativos de desempeño, aplicados a modelos tomados de la literatura abierta. Para finalizar el capítulo, en la Sección 4.5 se ofrecen conclusiones detalladas acerca de las implicancias del aporte realizado, el cual trasciende el área de aplicación de redes de datos constituyendo un avance generalizado para la disciplina de modelado y simulación.

El Capítulo 5 presenta aportes en el área de la implementación de modelos DEVS en tiempo real. Se desarrollan las herramientas necesarias para obtener un procedimiento completamente basado en modelos DEVS, los cuales puedan ser embebidos en un hardware de destino final, ejecutados en tiempo real, e interconectados con unidades externas hardware. La estrategia mantiene la continuidad de los modelos DEVS, al encapsular las funcionalidades de interacción con el hardware externo en modelos atómicos especiales capaces de interactuar con interfaces de control propias de cada dispositivo. En la Sección 5.1 se realiza un repaso de antecedentes en cuanto a formalismos y herramientas que facilitan la transición de modelos de sistemas hacia su implementación en hardware en tiempo real. En la Sección 5.2 se analiza el contexto particular de un procesador de red Intel IXP2400 como plataforma de ensayo y validación para ejecutar modelos DEVS en tiempo real controlando tráfico de red. Inmediatamente en la Sección 5.3 se presenta la adaptación realizada de la herramienta ECD++ haciéndolo capaz de ejecutar embebido en el hardware IXP2400. Se proveen nuevas bibliotecas para comunicar los modelos DEVS con capas de hardware específicas para manipulación de paquetes. Se describe también una nueva herramienta para automatizar la generación de las bibliotecas de bajo nivel que hace posible la integración entre ECD++ e IXP2400. Luego, la Sección 5.4 describe una serie de herramientas avanzadas para asistir al proceso de diseño de sistemas embebidos para control de redes: se desarrolla una interfaz visual para modelado y simulación con ECD++ y se implementa un laboratorio virtualizado y portable para desarrollar modelos DEVS con una placa de red de alta capacidad basada en IXP2400. La sección concluye con un ejemplo práctico desarrollando un control supervisorio de calidad de servicio de red utilizando las nuevas herramientas y realizando la validación con tráfico real. El capítulo cierra con algunas conslusiones en la Sección 5.5.

Casos de Aplicación

El Capítulo refcap:ctrldeadmision presenta una metodología integradora de modelado y simulación basada en DEVS para diseñar e implementar un sistema de control de calidad de servicio de redes, aplicando Teoría de Control e implementándolo embebido en un procesador de red. La Sección 6.1 presenta los sistemas de tiempo real que aplican teoría clásica de control a sistemas de cómputo y/o de red, los cuales son considerados como los objetos del control. La Sección 6.2 describe los antecedentes más importantes de sistemas de Control de Admisión (AC) mientras que la Sección 6.3 presenta un caso de estudio para obtener un AC aplicando Teoría de Control para obtener un control Proporcional–Integral (PI). Luego, la Sección 6.4 muestra el uso de la metodología propuesta basada en DEVS para asistir el proceso de análisis, diseño, verificación, implementación y validación del controlador PI especificado para el sistema. Primero se analiza y diseña el sistema con un modelo completamente continuo (ecuaciones diferenciales) tando de la red como del controlador propuesto. Luego se discretiza temporalmente el control, manteniendo la red en su aproximación fluida, con el objetivo de implementar posteriormente al controlador en su forma digital (discreto). Luego, con un propósito de verificación, se reemplaza la aproximación fluida de la red por su especificación a eventos discretos, acercándose más al sistema real. Finalmente, se hace referencia a ensayos realizados para traducir semiautomáticamente el modelo del controlador PI discreto obtenido con PowerDEVS hacia su versión ejecutable en ECD++ embebido en el procesador de red IXP2400. Se cierra con unas conclusiones en la Sección 6.5.

Un segundo caso de aplicación integrador es desarrollado en el Capítulo 7. En el mismo se muestra la utilización de PowerDEVS con los nuevos métodos DQSS para modelar una representación fluida del control de congestión tipo TCP/AQM. En la Sección 7.1 se especifica detalladamente el comportamiento esperado para los mecanismos de ventana deslizante (mecanismo Incremento Aditivo, Decremento Multiplicativo (AIMD)) y de detección temprana aleatoria (mecanismo RED), que operan en los nodos extremos comunicados y en el enrutador que los conecta, respectivamente. En la Sección 7.3 se desarrolla la aproximación fluida de TCP/AQM permitiendo simular en tiempos muy convenientes las características dinámicas más relevantes del sistema. Luego, continuando con PowerDEVS, se desarrollan modelos de eventos discretos para simular de manera exacta y detallada el comportamiento de la red bajo el esquema TCP/AQM, usando esta vez primeros principios de los protocolos y medios físicos. Los resultados obtenidos en este segundo paso permiten verificar la aproximación fluida realizada inicialmente, como así también estudiar detalladamente propiedades discretas más cercanas al sistema real. Luego, en la Sección 7.4 se implementa una estrategia de modelado de flujos híbridos para combinar la ventaja de la eficiencia computacional de una simulación fluída con la ventaja del detalle granular provisto por una simulación discreta. Mediante DEVS se obtienen flujos híbridos bajo un formalismo matemático unificado y en una misma herramienta. Se evidencia aquí la ventaja de mantener detalles granulares a nivel de paquetes, a la vez que se disminuyen sensiblemente los tiempos de simulación respecto de los obtenidos con el modelo eventos discretos puros. Se cierra el capítulo proveyendo conclusiones en la Sección 7.5.

Cierre

La presente Tesis finaliza en el Capítulo 8 con una discusión sobre los problemas abiertos que deja planteados este trabajo a modo de próximos pasos y las conclusiones generales.

1.2. Trabajos Relacionados

Existen algunos pocos antecedentes de intentos por relacionar formalmente a DEVS con sistemas estocásticos. En [Agg75, Zei76] los autores muestran que las simulaciones de eventos discretos que avanzan acorde a secuencias pseudo–aleatorias definen un modelo DEVS equivalente. En otras palabras, el formalismo DEVS puede capturar el comportamiento de sistemas estocásticos que son simulados utilizando secuencias pseudo–aleatorias, es decir, que ejecutan un *modelo determinista de un sistema estocástico* [Zei76]. En [Mel76] se establece una relación entre resultados de experimentos aleatorios y la evolución temporal de una simulación DEVS, al asignar medidas de probabilidad a las trayectorias de estados que son observables a nivel de entradas y salidas. Finalmente en [Jos96] se plantea un enfoque estructural para tomar en cuenta el comportamiento estocástico interno a nivel de transiciones de estados, limitado a tratar modelos DEVS con conjuntos de estados finitos. Los trabajos mencionados redundaron en un conjunto de resultados útiles que han sido suficientes para enfrentar algunos dominios de problema puntuales, pero únicamente desde una perspectiva comportamental, no estructural ni completamente genérica.

Con respecto a la simulación numérica de modelos descritos por conjuntos de Ecuaciones Diferenciales con Retardo (DDE) se encuentra cubierta por muy pocas publicaciones, y también, el estado del arte del software correspondiente está mucho menos desarrollado que el caso de ODE que ha sido de gran interés durante décadas. Muchos sistemas físicos pueden se aproximados por conjuntos de Ecuaciones Diferenciales Ordinarias (ODE), y consecuentemente, la simulación digital de dichos modelos ha captado el interés de ingenieros y matemáticos aplicados desde la invención de la computadora digital. Estos esfuerzos se encuentran resumidos en varios libros de texto [But87, CK06, HNW00, HW91, Lam91]. Sin embargo, existe un número significativo de sistemas en ciencias e ingenierías que requieren, en sus modelos, la inclusión de retardos [BPW95]. Existen buenos algoritmos de integración para DDE (Shampine, Thompson, y colaboradores) llamados dde23 [ST00, ST01] implementados en Matlab. Los autores también proveen un algoritmo en Fortran, llamado dde_solver [S.P08]. Ambos consisten en algoritmos clásicos de integración numérica en el sentido de que están basados en la idea clásica de la discretización del tiempo, utilizada en la mayoría de la literatura para ODE, DDE y Ecuaciones Diferenciales Algebráicas (DAE, por *Algebraic Differential Equations*).

DQSS por su parte, no cuenta con antecedentes en la literatura, constituyéndose en el primer algoritmo para integración numérica de DDE basados en la cuantificación de las variables de estado en lugar de la discretización del tiempo. Los algoritmos DQSS fueron implementados en la herramienta de modelado y simulación PowerDEVS [BK10], por medio de la cual se simplifica notablemente el uso de estos métodos de integración. DQSS extiende la familia de algoritmos de integración numérica basados en la cuantificación de estados

(QSS) para sistemas no–rígidos, descritos previamente en [CK06, KJ01].

En cuanto a los esfuerzos previos por brindar continuidad y consistencia de modelos hasta su implementación en sistemas embebidos en tiempo real, existen diversas líneas de trabajo que van desde lo académico hasta lo industrial. Por ejemplo BIP [ABS06] define componentes como la superposición de tres capas: Comportamiento, Interacciones y Prioridades, preservando propiedades durante la composición de modelos y soportando análisis y transformaciones entre límites heterogéneos (temporizado/no temporizado, síncrono/asíncrono, disparado por eventos/disparado por datos). Metropolis es un entorno para diseño de sistemas electrónicos que soporta especificación, simulación, análisis formal y síntesis. Se basa en metamodelos con semántica formal, capturando diseños a alto nivel de abstracción e implementando diferentes Modelos de Cómputo (MoC, por *Models of Computation*). Ptolemy II [EJL+03] permite el modelado jerárquico y estructurado de sistemas heterogéneos usando un Modelos de Cómputo (MoC) específico que contempla flujo de datos y flujo de control. ECSL (Embedded Control Systems Language) soporta el desarrollo de controladores distribuidos [BGK+06], incluyendo un entorno de dominio específico para sistemas automotrices. SystemC y Esterel son antecedentes de lenguajes a nivel de sistema, utilizados para simular y ejecutar modelos; y han sido beneficiados por una creciente adopción en la industria [BS08]. SystemC representa sistemas de hardware y software a diferentes niveles de abstracción, permitiendo elegir el nivel deseado para cada componente. Esterel se utiliza para sintetizar hardware y software por medio de un lenguaje basado en reacción y sentencias de alto nivel para manejar concurrencia. Otros esfuerzos importantes incluyen MoBIES (basado en comunicación de sistemas híbridos multi–agente), OCAPI-XL (focalizado en dispositivos en red con procesadores y aceleradores de hardware en un mismo chip), ForSyDe (basado en modelado de sistemas con MoCs heterogéneos, permitiendo simulación basada en Haskell) y Foresight (usando modelado y simulación basado en limitaciones de recursos, modelado gráfico, y generación de modelos ejecutables). Una de las técnicas más populares es UML-RT, la cual provee una metodología orientada a objetos. En [HS04] se provee una comparación entre UML-RT y DEVS, en la cual se muestra que a pesar de que el Profile UML-RT especifica tiempo, planificación, y performance usando objetos UML, los mismos no están formalmente definidos.

1.3. Resultados Originales

Se presentó un nuevo formalismo para describir sistemas de eventos discretos estocásticos basado en el enfoque de Teoría de Sistemas de DEVS, haciendo uso de la teoría de Espacios de Probabilidad. El nuevo formalismo, consistente en la especificación de Sistemas de Eventos Discretos Estocásticos (STDEVS, por *STochastic Discrete EVent Systems Specification*), provee un nuevo marco formal para modelado y simulación para sistemas de eventos discretos genéricos no deterministas. STDEVS provee la capacidad de representar comportamiento estocástico sobre espacios medibles de dimensión infinita, junto con la capacidad de representar formalismos estocásticos tradicionales vastamente utilizados en diversas aplicaciones prácticas (por ejemplo, Cadenas de Markov, Redes de Colas, Redes de Petri Estocásticas). Debido a su raíz basada en DEVS, STDEVS es también una representación basada en Teoría de Sistemas, proveyendo ventajas

específicas de modelado interdisciplinario para especificar sistemas híbridos.

Se desarrolló también una nueva clase de algoritmos de integración Ecuaciones Diferenciales con Retardo (DDE) y se mostró su aplicación por medio de un número de problemas de la literatura, estudiando su desempeño. Los nuevos métodos DQSS se basan en la cuantificación de las variables de estado en lugar de discretizar el tiempo, ofreciendo algunas ventajas muy interesantes que los hace particularmente atractivos. Se mostró por medio de cuatro problemas de análisis de desempeño distintos que la solución numérica de DDE por cuantificación de estados es frecuentemente significativamente más eficiente que la estrategia basada en discretización del tiempo. La razón para esta mayor eficiencia radica parcialmente en que DQSS explota naturalmente la ralitud de los modelos a simular, y parcialmente en que los algoritmos utilizados para interpolar los estados retardados son muy sencillos y pueden así ser implementados más eficientemente. El código de DQSS implementado en PowerDEVS es mucho más sencillo de utilizar que otras alternativas populares como dde23 y que dde_solver, gracias a la interfaz de usuario gráfica intuitiva que provee PowerDEVS (similar a la provista por Simulink).

Estas herramientas trascienden el dominio de aplicación de las redes de datos constituyéndose en soluciones generalizadas para la disciplina de modelado y simulación.

También se diseñaron un nuevo procedimiento y conjunto de herramientas basados completamente en modelos DEVS, para que los mismos puedan ser a) embebidos en un hardware de destino final, b) ejecutados en tiempo real y c) conectados de forma transparente a unidades externas hardware (Hardware–In–the–Loop, HIL) con capacidad de acción y reacción. La estrategia mantiene la continuidad de los modelos DEVS, al encapsular las funcionalidades de interacción con el hardware externo en modelos atómicos especiales (Mappers) capaces de interactuar con interfaces de control (Drivers) propias de cada dispositivo. Se realizó una integración de la herramienta Embedded CD++ con un procesador de red Intel IXP2400 y se desarrollaron bibliotecas de interfaz para la operación tipo HIL entre el simulador y los dispositivos de bajo nivel de dicho procesador. También se desarrollaron nuevas herramientas visuales avanzadas para simplificar y automatizar el desarrollo de modelos ECD++ y de las bibliotecas de interfaz mencionadas. Adicionalmente se integraron múltiples herramientas de desarrollo necesarias para la experimentación con ECD++ y la placa de red RadiSys ENP2611 (basada en un IXP2400), creando un laboratorio virtualizado portable que simplifica notablemente las tareas de validación e implementación de modelos.

Al desarrollar los casos de estudio prácticos de los capítulos 5, 6 y 7 se introducen diversos modelos originales que quedan disponibles en las bibliotecas de acceso público de PowerDEVS y ECD++. Se destacan entre ellos el modelo generador de paquetes, cola, servidor, medidor de tasa, balanceador de carga, compuertas de admisión tipo Token Bucket y Random Early Detection y canal de ancho de banda y latencia finitos, entre otros. Se destaca el modelo discreto de control de flujo tipo TCP, que representa el primero en su clase desarrollado en DEVS con un grado de detalle considerable.

1.4. Publicaciones de Respaldo

La idea original de STDEVS fue publicada en una conferencia local en [KC06], y posteriormente reformulada y publicada en un congreso internacional [CKW08]. Finalmente resultó en un artículo publicado en la revista internacional de referencia en la disciplina [CKW10].

El artículo sobre los métodos de DQSS fue publicado en una de las revistas internacionales más importantes de la disciplina [CKC10], constituyendo el primer método de integración numérica por cuantificación de estados para ecuaciones diferenciales con retardos.

Por su parte, la publicación [BWC10a] y su correspondiente versión extendida [BWC12] presentan un nuevo entorno gráfico para desarrollo de modelos con CD++.

A modo de aportes colaterales en revistas y congresos que respaldan las líneas de trabajo de la presente Tesis se pueden mencionar las siguientes publicaciones.

En [JWBC09] se presenta por primera vez la implementación de ECD++ embebido en el procesador de red INTEL IXP2400 y se proponen controles supervisorios de calidad de servicio de tráfico basados en DEVS.

En [CKW09] se describe la metodología integral basada completamente en DEVS que permite modelar controles para redes de datos basados en Teoría de Control, realizar transformaciones de paradigma de modelado, y finalmente transicionar hacia la implementación en tiempo real.

El artículo de revista [WC09] realiza un raconto de CD++ y su utilización en diversas aplicaciones utilizando el formalismo CellDEVS para especificar autómatas celulares con DEVS.

Capítulo 2

Conceptos Preliminares

Como se mencionó en la introducción, las redes de datos pueden modelarse con diversos paradigmas de acuerdo a las necesidades, conduciendo en general a especificaciones híbridas.

En el presente capítulo se proveerán los conceptos preliminares necesarios para comprender las características de los *Sistemas Dinámicos Híbridos* (Sección 2.1) para caracterizar las principales representaciones a las que puede recurrirse en el modelado de redes de datos. Allí se pondrá de manifesto la problemática de la interacción de las variables discretas y continuas, produciendo discontinuidades en los métodos clásicos de integración numérica. Se presentarán los Sistemas de Estados Cuantificados (QSS, por *Quantized State Systems*) que reemplazan la discretización temporal por la cuantificación de las variables de estado, y de las Ecuaciones Diferenciales con Retardo (DDE, por *Delay Differential Equations*) en las cuales la dinámica de un sistema depende tanto de su estado actual como de su historia. Los QSS y las DDE tendrán un rol central en el desarrollo de nuevas herramientas para el modelado continuo de redes.

Se postulará luego a DEVS como formalismo de modelado y simulación integrador para sistemas dinámicos híbridos en general, pasando inmediatamente a presentar en mayor detalle al formalismo DEVS y los QSS en las Secciones 2.2 y 2.3 respectivamente.

El capítulo prosigue con la Sección 2.4 que presentará dos herramientas de software basadas en DEVS para modelado y simulación de sistemas híbridos: PowerDEVS y ECD++. Estas herramientas de software servirán de soporte para la implementación, verificación y validación de los aportes realizados a lo largo de la Tesis.

El capítulo concluye en la Sección 2.5 en donde se pondrá en contexto a DEVS con el modelado de redes de datos, y se proveerá un escenario de estudio motivador para identificar necesidades no cubiertas para poder aplicar DEVS como formalismo integrador. El escenario de estudio motivador acompañará el desarrollo de la Tesis tanto desde el punto de vista de los aportes teóricos como de las herramientas prácticas que se desarrollarán.

2.1. Sistemas Dinámicos Híbridos

La Teoría de Sistemas suele dividir los sistemas dinámicos en tres grandes categorías acorde a la forma en que sus variables descriptivas evolucionan en el tiempo: Sistemas de Tiempo Continuo (cambian continuamente con el tiempo), de Tiempo Discreto (cambian en instantes determinados de tiempo) y de Eventos Discretos (pueden cambiar en cualquier instante, pero sólo un número finito de veces dado un intervalo acotado). La presencia o no de componentes de evolución aleatoria subdivide las categorías anteriores entre modelos estocásticos y deterministas.

Mientras los sistemas discretos se pueden simular de manera directa, los sistemas de tiempo continuo requieren de la utilización de *métodos de integración numérica*, que generalmente aproximan las ecuaciones diferenciales mediante ecuaciones en diferencias discretizando la variable independiente (el tiempo). La integración numérica de ecuaciones diferenciales es una disciplina en sí misma y la literatura da cuenta de cientos de algoritmos basados en la discretización temporal [But87, Lam91, HNW00, HW91, CK06].

Muchos sistemas de la ingeniería combinan características continuas y discretas. Estos sistemas se denominan *híbridos* y sus modelos matemáticos contienen ecuaciones diferenciales cuyas variables interactúan con otras variables de evolución discreta.

La interacción entre las variables continuas y discretas de los sistemas híbridos provoca discontinuidades en las ecuaciones diferenciales. Estas discontinuidades causan muchas dificultades a los algoritmos de integración numérica ya que los mismos deben detectar exactamente los instantes en los que se producen y reiniciar la simulación a partir de dichos instantes [CK06], lo que conlleva un aumento muy importante de los costos computacionales.

Un problema similar experimentan los métodos de integración en presencia de Ecuaciones Diferenciales con Retardo (DDE). En estos sistemas la evolución de las variables depende tanto de sus valores actuales como de su historia, con la particularidad de que la presencia de discontinuidades en las derivadas es un fenómeno intrínseco al sistema. Esto ha motivado el desarrollo de algoritmos numéricos especiales [WB92, ST01]. Los sistemas híbridos con retardos, a su vez, combinan ambas dificultades [S.P08].

Un enfoque esencialmente distinto al de los métodos *clásicos* de integración reemplaza la discretización temporal por la cuantificación de las variables de estado, dando lugar a los llamados métodos de *integración por cuantificación de estados*. Estos métodos se basan en una idea original de Bernard Zeigler, quien propuso que los sistemas continuos podían aproximarse por sistemas de eventos discretos bajo el formalismo DEVS [ZKP00]. Posteriormente, en [KJ01] se demostró que la cuantificación debía realizarse con el agregado de histéresis y con esta idea se formalizó el primer método de integración por cuantificación llamado QSS. Actualmente existen varios métodos de integración por cuantificación: QSS1, QSS2 [Kof02, CK06], QSS3 [Kof06] y los métodos para sistemas rígidos BQSS, LIQSS1, LIQSS2 y LIQSS3 [MK09, Mig10].

Los QSS tienen propiedades teóricas muy fuertes: convergencia, estabilidad [KJ01, CK06] y una propiedad notable de existencia de cota de error global calculable [Kof02, CK06]. La característica asíncrona de los métodos QSS los hace particularmente eficientes para la integración numérica de sistemas con discontinuidades [Kof04], donde pueden reducir en más de un orden de mag-

nitud los costos computacionales respecto de los métodos clásicos. Por razones similares, los métodos QSS son también muy eficientes para simular ecuaciones diferenciales con retardos [CKC10a]. Dado un sistema continuo (o híbrido, con o sin retardos), el uso de cualquier QSS lo aproxima mediante un sistema de eventos discretos DEVS.

El formalismo para la Especificación de Sistemas de Eventos Discretos (DEVS) fue propuesto a mediados de los años setenta [Zei76, ZKP00] como una metodología genérica para describir sistemas de eventos discretos.

DEVS es una representación basada en la Teoría de Sistemas para describir sistemas cuyo comportamiento puede ser expresado por medio de secuencias de eventos.

Al ser un formalismo universal para sistemas de eventos discretos, otras metodologías (p.ej. Finite State Automata, Petri Nets, Grafcets, Statecharts, etc.) pueden ser representados por DEVS. Esta generalidad convirtió a DEVS en un formalismo ampliamente utilizado para describir y simular la mayoría de los sistemas discretos, incluyendo sistemas de tiempo discreto [ZKP00].

Aún más, se han desarrollado recientemente métodos de integración numérica que aproximan sistemas continuos (ecuaciones diferenciales) por medio de modelos DEVS [CK06] y varias propuestas de extensiones al formalismo DEVS original para el modelado y simulación de sistemas continuos e híbridos [GEG00, Nut03]. Acompañando al desarrollo teórico, una considerable variedad de herramientas de modelado y simulación basadas en DEVS viene siendo desarrollada durante la última década [ZS00, WCD01, FDB02, KLP03].

2.2. Formalismo DEVS

La capacidad de DEVS para describir modelos híbridos generalizados se logra gracias a su capacidad de representar cualquier sistema de eventos discretos, es decir, cualquier sistema cuyo comportamiento de entrada/salida pueda ser descrita por secuencias de eventos.

Específicamente, un modelo DEVS [ZKP00] procesa una trayectoria de eventos de entrada y –acorde a dicha trayectoria y a su propio estado inicial– produce una trayectoria de eventos de salida. Este comportamiento se muestra en la Fig.2.1.

Figura 2.1: Comportamiento Entrada/Salida de un modelo DEVS.

Un *evento* es la representación de un cambio instantáneo en alguna parte del sistema. Como tal, puede caracterizarse mediante un valor y un tiempo de ocurrencia. El valor puede ser un número, un vector, una palabra, o en general, un elemento de un conjunto arbitrario.

Una *trayectoria* queda definida por una secuencia de eventos. La misma toma el valor ϕ (o *No Event*) en casi todos los instantes de tiempo, excepto en los instantes en los que sí hay eventos. En estos instantes, la trayectoria toma el valor correspondiente al evento. La Figura 2.2 muestra una trayectoria de eventos que toma los valores x_2 en el tiempo t_1, luego toma el valor x_3 en t_2, etc.

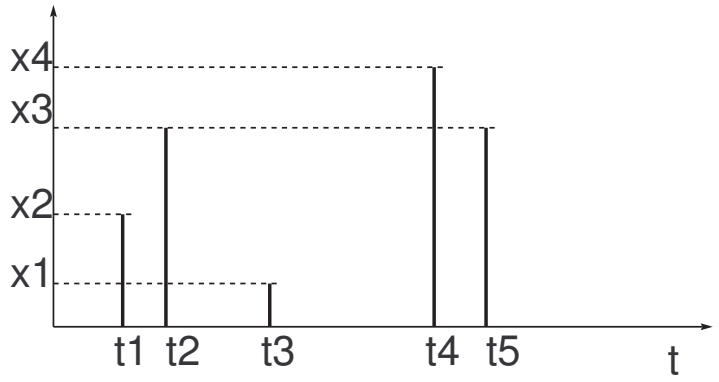


Figura 2.2: Ejemplo de una trayectoria genérica de eventos discretos.

2.2.1. Modelos DEVS Atómicos

Formalmente un modelo atómico DEVS se define con la siguiente estructura:

$$M = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

donde

- X es el conjunto de eventos de entrada, es decir, el conjunto de todos los valores que puede tomar un evento de entrada;
- Y es el conjunto de eventos de salida;
- S es el conjunto de los estados internos;
- δ_{int}, δ_{ext}, λ y ta son funciones que definen la dinámica (comportamiento) del modelo.

Cada posible estado s ($s \in S$) tiene asociado un *avance de tiempo* calculado por la *función de avance de tiempo* $ta(s)$ ($ta(s) : S \to \Re_0^+$). El *avance de tiempo* es un número real no negativo que indica cuánto tiempo debe permanecer el modelo en un estado dado, en ausencia de eventos de entrada.

Luego, si el estado asume el valor s_1 en el instante t_1, luego de $ta(s_1)$ unidades de tiempo (es decir, en el instante $ta(s_1) + t_1$) el sistema ejecuta una *transición interna*, pasando al nuevo estado s_2. Dicho estado es calculado como $s_2 = \delta_{\text{int}}(s_1)$, donde δ_{int} ($\delta_{\text{int}} : S \to S$) se llama *función de transición interna*.

Cuando el estado pasa de s_1 a s_2 se emite un evento de salida con valor con valor $y_1 = \lambda(s_1)$, donde λ ($\lambda : S \to Y$) se denomina *función de salida*. Las funciones ta, δ_{int}, y λ definen completamente el comportamiento autónomo de un modelo DEVS.

Cuando llega un evento de entrada, el estado cambia instantáneamente. El nuevo valor del estado depende no sólo del valor del evento de entrada, sino también del valor del estado anterior y del tiempo transcurrido desde la última transición. Si el sistema pasa a un nuevo estado s_3 en el instante t_3 y luego llega un evento de entrada en el instante $t_3 + e$ con valor x_1, el nuevo estado se calcula como $s_4 = \delta_{\text{ext}}(s_3, e, x_1)$ (notar que $ta(s_3) \geq e$). En este caso, se dice que el sistema ejecuta una *transición externa*.

La función δ_{ext} ($\delta_{\text{ext}} : S \times \Re_0^+ \times X \to S$) se denomina *función de transición externa*. Durante una transición externa no se produce ningún evento de salida.

La Figura 2.3 muestra el comportamiento típico de un modelo DEVS.

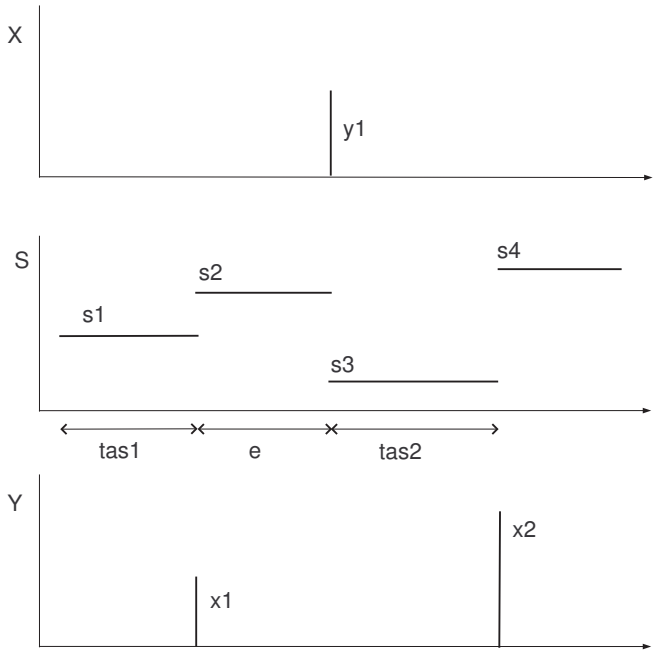


Figura 2.3: Trayectorias de eventos y estados en un modelo DEVS.

Los conjuntos X e Y de donde toman los valores de eventos de entrada y salida pueden formarse por el producto cartesiano de un conjunto arbitrario de valores de eventos y un conjunto que contenga un número finito de *puertos* de entrada y salida, respectivamente.

Sea por ejemplo un sistema que calcula una función estática $f(u_0, u_1)$, donde u_0 y u_1 son trayectorias seccionalmente constantes.

Una trayectoria seccionalmente constante puede ser representada mediante una secuencia de eventos si se relaciona cada evento con un cambio en el valor de la trayectoria. Utilizando esta idea, se puede construir el siguiente modelo DEVS atómico:

$$
\begin{aligned}
&M_F = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta), \text{ donde} \\
&X = \Re \times \{0, 1\} \\
&Y = \Re \times \{0\} \\
&S = \Re^2 \times \Re_0^+ \\
&\delta_{\text{int}}(s) = \delta_{\text{int}}(u_0, u_1, \sigma) = (u_0, u_1, \infty) \\
&\delta_{\text{ext}}(s, e, x) = \delta_{\text{ext}}(u_0, u_1, \sigma, e, x_v, p) = \tilde{s} \\
&\lambda(s) = \lambda(u_0, u_1, \sigma) = (f(u_0, u_1), 0) \\
&ta(s) = ta(u_0, u_1, \sigma) = \sigma
\end{aligned}
$$

con:

$$
\tilde{s} = \begin{cases} (x_v, u_1, 0) & \text{si } p = 0 \\ (u_0, x_v, 0) & \text{en otro caso} \end{cases}
$$

En este modelo, se incluyó una variable σ en el estado s que coincide con la función ta. Esto se suele hacer siempre, ya que facilita la obtención del modelo DEVS atómico.

Como se dijo anteriormente, cada evento de entrada y de salida incluye un número entero que indica el correspondiente puerto de entrada o salida. En los eventos de entrada, el puerto p puede ser 0 o 1 (es decir, hay dos puertos de entrada, uno para u_0y el otro para u_1). En los eventos de salida, hay un sólo puerto de salida (0).

2.2.2. Modelos DEVS Acoplados

Dado que DEVS es un formalismo muy general suele ser atractivo para describir sistemas heterogéneos y complejos. Sin embargo, representar un sistema muy complejo usando exclusivamente las funciones matemáticas de transición y avance de tiempo es una tarea muy difícil, ya que se deberían tener en cuenta en forma simultánea todos los posibles estados del sistema. Esto suele ser engorroso aún en sistemas de complejidad modesta.

Los sistemas complejos generalmente se piensan como el acoplamiento de sistemas más simples. A través del acoplamiento, los eventos de salida de unos subsistemas se convierten en eventos de entrada de otros subsistemas. DEVS garantiza que el modelo resultante de acoplar varios modelos DEVS atómicos es equivalente a un nuevo modelo DEVS atómico, es decir, DEVS es cerrado frente al acoplamiento (clausura del acoplamiento). Esto permite el acoplamiento jerárquico de modelos DEVS, o sea, la utilización de modelos acoplados como si fueran modelos atómicos que a su vez pueden acoplarse con otros modelos atómicos o acoplados.

Una forma de acoplamiento muy práctica es el acoplamiento mediante puertos de entrada/salida.

La Figura 2.4 muestra un modelo DEVS acoplado N, resultado de acoplar los modelos M_a y M_b. De acuerdo a la propiedad de clausura de DEVS, el modelo N puede usarse de la misma forma que si fuera un modelo atómico, y puede ser acoplado con otros modelos atómicos y/o acoplados.

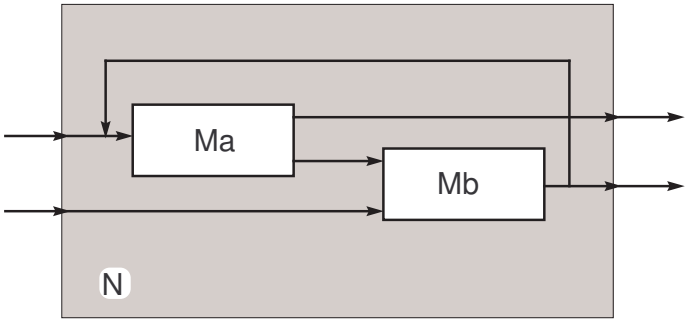


Figura 2.4: Modelo DEVS acoplado.

Para poder mostrar las conexiones entre los distintos modelos DEVS acoplados, en esta Tesis se utilizarán *Diagramas de Bloques* similares al mostrado en la Figura 2.4.

Un modelo DEVS Acoplado N se define formalmente por la siguiente estructura:

$$N = (X_N, Y_N, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\}, Select)$$

donde:

- X_N y Y_N son los conjuntos de calores de entrada y salida del modelo acoplado.
- D es el conjunto de referencias a los componentes internos, de modo que cada $d \in D$, M_d es un modelo DEVS.
- Para cada $d \in D \cup \{N\}$, $I_d \subset (D \cup \{N\}) - \{d\}$ es el conjunto de modelos influyentes sobre el subsistema d.
- Para cada $i \in I_d$, $Z_{i,d}$ es la función de traducción, donde

$$Z_{i,d} : \begin{cases} X_N \to X_d & \text{if } i = N \\ Y_i \to Y_N & \text{if } d = N \\ Y_i \to X_d & \text{en cualquier otro caso} \end{cases}$$

- $Select : 2^D \to D$ es una función de desempate para eventos simultáneos, y debe cumplir $Select(E) \in E$, con $E \subset 2^D$ siendo 2^D el sub conjunto de componentes que produce la simultaneidad de eventos.

2.3. Métodos de Integración por Cuantificación de Estados

Como se mencionó antes, mientras DEVS puede modelar y simular cualquier sistema discreto (de tiempo discreto y de eventos discretos), los sistemas continuos no pueden ser representados directamente debido a la evolución continua de sus variables.

A pesar de que las ecuaciones diferenciales pueden modelar sistemas continuos, para simularlas se requieren métodos de integración numérica (p.ej., Euler o Runge Kutta) obteniendo las ecuaciones en diferencia aproximantes. Luego, estos modelos de tiempo discreto son representables de modo sencillo con DEVS. Siguiendo esta idea, se pueden simular sistemas híbridos genéricos bajo un formalismo unificado.

Los métodos clásicos de integración numérica tienen problemas con las discontinuidades presentes en los sistemas híbridos (causados por las dinámicas discretas). Estas deben ser detectadas y tratadas debidamente ya que de otro modo la aproximación de las porciones continuas puede conducir a resultados incorrectos [CK06]. La detección y tratamiento apropiados discontinuidades es técnicamente difícil y computacionalmente muy demandante ya que requiere iteraciones. DEVS, en cambio, puede representar aproximaciones de tiempo discreto y eventos discretos para sistemas continuos. Los métodos QSS hacen posible aproximar una ecuación diferencial por un modelo de eventos discretos preservando propiedades de estabilidad y garantizando cotas de error. QSS puede detectar y tratar discontinuidades de un modo eficiente y sencillo sin recurrir a iteraciones [CK06]. Por ello, son particularmente convenientes para simular sistemas híbridos.

En esta sección se presentarán los métodos de *integración numérica por cuantificación de estados* conocidos como Sistemas de Estados Cuantificados (QSS). En primer lugar se presentará la idea básica y las características principales de los métodos QSS aplicados a Ecuaciones Diferenciales Ordinarias (ODE, por *Ordinary Differential Equations*). Luego, se mostrará que los sistemas aproximados por QSS pueden ser representados en el marco del formalismo DEVS.

2.3.1. Métodos QSS para ODE

Considérese un conjunto de ODE invariantes en el tiempo representado en su forma de espacio de estados:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{2.1}$$

donde $\mathbf{x}(t) \in \Re^n$ es el vector de estado, y $\mathbf{u}(t) \in \Re^w$ es un vector de trayectorias de entrada, conocidas y constantes a tramos.

El método QSS de primer orden (QSS1) simula una versión aproximada de la Ec. (2.1), y se lo denomina Quantized State System (QSS):

$$\dot{\mathbf{x}}(t) = f(\mathbf{q}(t), \mathbf{u}(t)) \tag{2.2}$$

donde $\mathbf{q}(t)$ es un vector de estados cuantificados que resulta de la cuantificación de las variables de estado $x_j(t)$.

Cada componente de $\mathbf{q}(t)$ sigue una trayectoria constante a tramos, relacionada con su correspondiente componente de $\mathbf{x}(t)$ por una función de cuantificación con histéresis definida por:

$$q_j(t) = \begin{cases} x_j(t) & \text{if } |q_j(t^-) - x_j(t)| = \Delta Q_j \\ q_j(t^-) & \text{en cualquier otro caso} \end{cases} \tag{2.3}$$

con $q_j(t_0) = x_j(t_0)$ y siendo t^- el límite por izquierda del instante t. Luego, $q_j(t)$ resulta en una aproximación constante a tramos de $x_j(t)$ que cambia su valor

únicamente cuando ambas trayectorias difieren en $\pm\Delta Q_j$. La magnitud ΔQ_j se denomina *quantum*. La relación entre x_j y q_j se describe en la Figura 2.5.

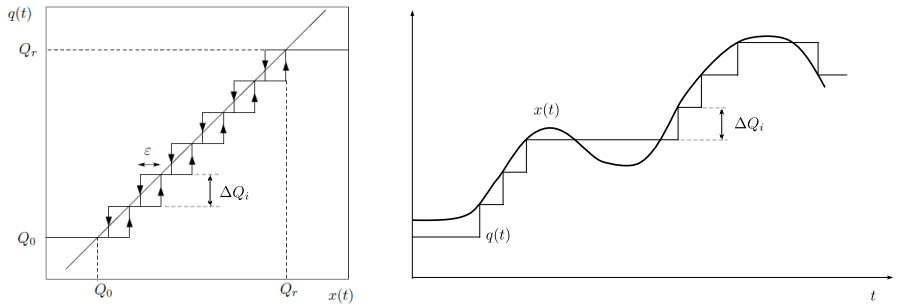


Figura 2.5: Función de cuantificación de estado con histéresis (izquierda). Cuantificación con ancho de histéresis $\varepsilon = \Delta Q$ (derecha)

En la Figura 2.5 (derecha), se muestra un ejemplo de cuantificación para una trayectoria arbitraria $x(t)$ aplicando la relación de Ec. (2.3). La presencia de histéresis en QSS1 evita la presencia de oscilaciones de frecuencia infinita que impedirían el avance del tiempo de simulación [CK06].

El método QSS1 puede simular *cualquier* sistema con la estructura de la Ec. (2.1), ofreciendo las siguientes propiedades:

- Preserva la estabilidad numérica (sin involucrar fórmulas implícitas). La cuantificación puede ser tratada cono una perturbación acotada de la ODE original, de modo que la estabilidad no lineal puede ser estudiada por medio de funciones de Lyapunov [KJ01].
- Ofrece una cota de error global, que garantiza que la solución numérica de un sistema lineal invariante en el tiempo analíticamente estable nunca diferirá de su solución analítica por una cantidad mayor a un valor finito y calculable [CK06].
- Es intrínsecamente asíncrono: cada variable de estado actualiza su valor independientemente del resto de los estados, en distintos instantes. Esto ofrece una ventaja significativa en términos de performance cuando se tratan sistemas con matrices ralas.
- Provee salida densa, una característica particularmente útil para métodos asíncronos.
- Es muy eficiente al simular a través de discontinuidades fuertes, debido a la simplicidad de los procedimientos requeridos para resolver raíces de polinomios cuando se dispone de una salida densa. Asimismo, debido a las propiedades asíncronas del método, cada discontinuidad puede manejarse eficientemente de forma aislada.

Mientras los algoritmos de integración clásicos para ODE conducen a aproximaciones de tiempo discreto, es decir, a conjuntos de ecuaciones en diferencia,

los métodos QSS no lo hacen. En cambio, la aproximación QSS de la Ec. (2.2) resulta en un sistema de eventos discretos, que puede modelarse usando el formalismo DEVS [Zei76, ZKP00].

2.3.2. Relación entre QSS y el formalismo DEVS

La Figura 2.6 muestra un diagrama de bloques de la Ec. (2.2). En gris, la figura muestra dos tipos de subsistemas: bloques tipo F_i (funciones estáticas) y bloques tipo HQI (integradores cuantificados con histéresis). Los bloques F_i corresponden a las funciones que calculan la parte a la derecha de la Ec. (2.2), mientras que HQI se compone de un integrador y una función de cuantificación con histéresis.

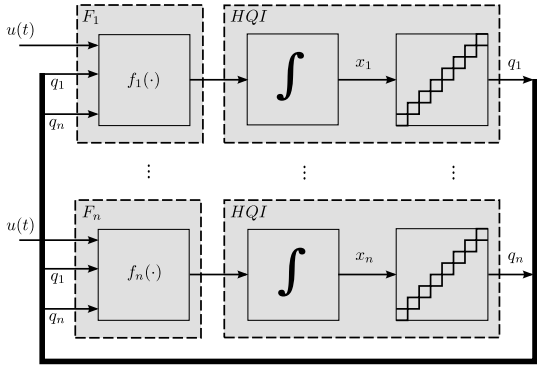


Figura 2.6: Representación de QSS mediante diagrama de bloques. Referencias de flechas: fina (señal simple), gruesa (bus de múltiples señales).

Cada bloque F_i tiene a su entrada trayectorias continuas a tramos q_j y u_j, y calcula una trayectoria de salida continua a tramos $\dot{x}_j$. Similarmente, los bloques HQI reciben entradas continuas a tramos $\dot{x}_j$, y calculan sus trayectorias de salida continuas a tramos q_j.

Ya que las trayectorias continuas a tramos pueden representarse por secuencias de eventos, y recordando que los modelos atómicos DEVS pueden representar cualquier sistema que reciba y/o produzca este tipo de secuencias, cada bloque en la Fig.2.6 puede ser representada por un modelo atómico DEVS.

Para las funciones estáticas F_i y los integradores cuantificados con histéresis HQI los modelos DEVS equivalentes resultan ser muy sencillos, y se encuentran especificados en [CK06].

Luego, puede obtenerse un modelo DEVS equivalente a Ec. (2.2) al acoplar los modelos DEVS atómicos correspondientes a F_i y a HQI del modo que se muestra en el diagrama de bloques de la Fig.2.6.

2.3.3. QSS de Órdenes Superiores

El mecanismo QSS1 descrito hasta aquí es un algoritmo de aproximación de primer orden que ofrece una relación lineal entre la precisión y el número de pasos requeridos para la simular un sistema dado. Luego, exhibe un desempeño pobre para los niveles de precisión relativamente altos requeridos para la mayoría de las aplicaciones ingenieriles.

Por este motivo, se han desarrollado métodos QSS de ordenes superiores, en particular, métodos QSS de segundo orden (QSS2), tercer orden (QSS3), y cuarto orden (QSS4). La idea básica es reemplazar la función de cuantificación que calcula $q_j(t)$ para preservar información de órdenes superiores sobre las señales originales $x_j(t)$.

En la cuantificación de QSS1, se preserva la primer derivada de $x_j(t)$, obteniendo para $q_j(t)$ una aproximación de constantes a tramos (tipo función escalera). En QSS2, se preserva la segunda derivada de $x_j(t)$, obteniendo para $q_j(t)$ una aproximación lineal a tramos (sucesión de rectas), mientras que en QSS3, la tercer derivada temporal de $x_j(t)$ es preservada, produciendo trayectorias parabólicas a tramos (sucesión de parábolas) en $q_j(t)$. Para QSS4, se preserva la cuarta derivada de $x_j(t)$ y se obtiene $q_j(t)$ como hipérbolas a tramos.

En la Figura 2.7 se muestra la cuantificación $q_j(t)$ para una señal de entrada arbitraria $x_i(t)$ using un quantum ΔQ_j, para los métodos QSS2 y QSS3.

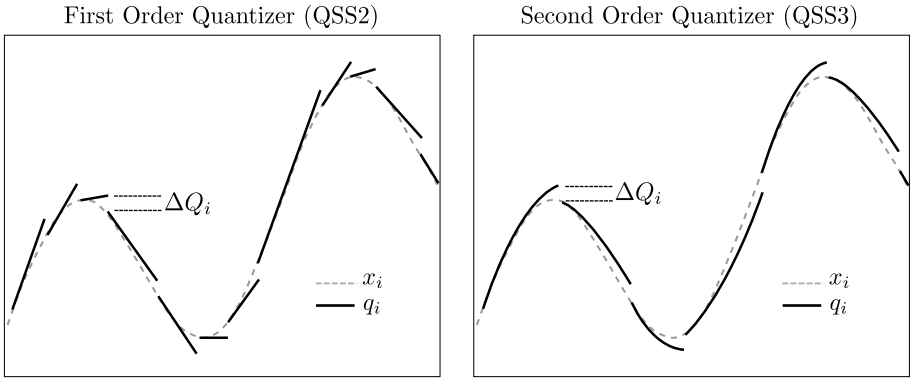


Figura 2.7: Cuantificación de señales. De primer orden (usada en QSS2, izquierda) y de segundo orden (usada QSS3, derecha)

La definición formal de los métodos QSS2, QSS3 y QSS4 es la misma que la de QSS1. Los integradores QSS2 y QSS3 aproximan al sistema ODE de la Ec. (2.1) con el sistema QSS de la Ec. (2.2), pero ahora las versiones cuantificadas de los estados q_j se calculan a partir de x_j como se muestra en la Fig.2.7.

En términos de los problemas de desempeño mencionados antes, los métodos QSS de orden superior ofrecen una propiedad muy importante: para QSS2, el número de pasos del algoritmo crece con la raíz cuadrada de la precisión, para QSS3, el número de pasos crece con la raíz cúbica de la precisión, y para QSS4, sucede lo propio con la raíz cuarta. Cabe destacar que QSS1, QSS2, QSS3 y QSS4 comparten las mismas propiedades teóricas (al menos para sistemas lineales).

La implementación DEVS de QSS2, QSS3 y QSS4 es análoga a la de QSS1. Siempre se sigue la estructura del diagrama de bloques de la Fig.2.6, pero cambian los modelos atómicos para las funciones estáticas F_i y los integradores cuantificados HQI, ya que deberán tener en cuenta las derivadas de orden superior de sus trayectorias de entrada y salida.

Por ejemplo, para QSS2, los bloques F_i y HQI procesan trayectorias de entrada y salida lineales a tramos. Como consecuencia, los modelos DEVS para F_i y HQI reciben y producen eventos que transportan la información de dos coeficientes polinomiales representando el valor y la pendiente (respectivamente)

de cada sección polinómica, que se mantiene válida entre evento y evento.

2.4. Herramientas de Simulación DEVS

Una de las características más importantes de DEVS es su facilidad para implementar simulaciones. Un modelo DEVS puede simularse con un programa ad–hoc escrito en cualquier lenguaje. De hecho, la simulación de un modelo DEVS no es mucho más complicada que la de un modelo de tiempo discreto.

Un algoritmo básico que puede utilizarse para simular un modelo DEVS acoplado puede describirse por los siguientes pasos:

1. Se identifica el modelo atómico que, de acuerdo a su tiempo de avance y al tiempo transcurrido, deba ser el próximo en realizar una transición interna. Se refiere con d^* a este modelo y t_n al tiempo de dicha transición.

2. Se avanza el reloj de la simulación t a $t = t_n$, y se ejecuta la función de transición interna del modelo d^*.

3. Se propaga el evento de salida provocado por d^* a todos los modelos atómicos conectados al puerto de salida y se ejecutan las funciones de transición externas correspondientes. Luego, se regresa al paso 1.

Aunque, la implementación de una simulación de DEVS es muy simple, en general los modelos que se utilizan en la práctica están compuestos por muchos subsistemas, con lo que hacer un programa ad–hoc de todos estos modelos suele tornarse muy trabajoso.

En esta Tesis se utilizarán dos herramientas de software

Para la mayor parte del presente trabajo se utilizará la herramienta denominada PowerDEVS para las fases de modelado, simulación, análisis y verificación de redes de datos. Este software, a pesar de ser un simulador de DEVS de propósito general, tiene particularidades específicas que facilitan la simulación de sistemas continuos.

Para la fase de simulación embebida en tiempo real de los modelos obtenidos y su posterior validación, se utilizará la herramienta ECD++. Este software permite la ejecución en tiempo real de modelos DEVS en plataformas de escasos recursos de hardware.

Ambas herramientas comparten la característica de estar construidas en C++, y que dicho lenguaje es la forma estándar para especificar las funciones dinámicas de los modelos DEVS atómicos.

2.4.1. PowerDEVS

PowerDEVS tiene una interfase gráfica que permite crear modelos DEVS acoplados con las herramientas típicas de drag–and–drop. Cuenta también con bibliotecas que tienen varios modelos atómicos ya definidos (muchos de ellos para realizar simulaciones con métodos de integración QSS).

Además, los modelos atómicos pueden crearse y modificarse fácilmente utilizando el *editor de modelos atómicos*, donde el usuario sólo debe definir las funciones de transición, de salida y de avance de tiempo utilizando sintaxis C++.

Toda la familia de métodos QSS (incluidos los nuevos métodos que se presentarán en esta Tesis) fueron implementados en PowerDEVS. Este software, a pesar de ser un simulador de DEVS de propósito general, fue concebido especialmente para facilitar la simulación de sistemas híbridos basada en métodos QSS [CK06]. PowerDEVS tiene una interfase gráfica similar a Simulink que permite editar diagrama de bloques. A un nivel inferior, cada bloque tiene una descripción en lenguaje C++ del atómico DEVS que puede ser editada (usando la herramienta para edición de atómicos de PowerDEVS). Además, la distribución del mismo cuenta con bibliotecas que tienen todos los bloques necesarios para modelar sistemas continuos e híbridos (integradores, funciones matemáticas, bloques para manejo de discontinuidades, fuentes, etc.). En consecuencia, un usuario puede modelar y simular usando los métodos QSS sin necesidad de saber nada de DEVS, simplemente como si modelara o simulara en Simulink.

En las figuras 2.8 y 2.9 pueden verse dos capturas de pantalla en las que se ve la biblioteca de bloques continuos de PowerDEVS y el modelo de un motor de corriente continua con un control de velocidad, junto a la interfaz de control de la simulación.

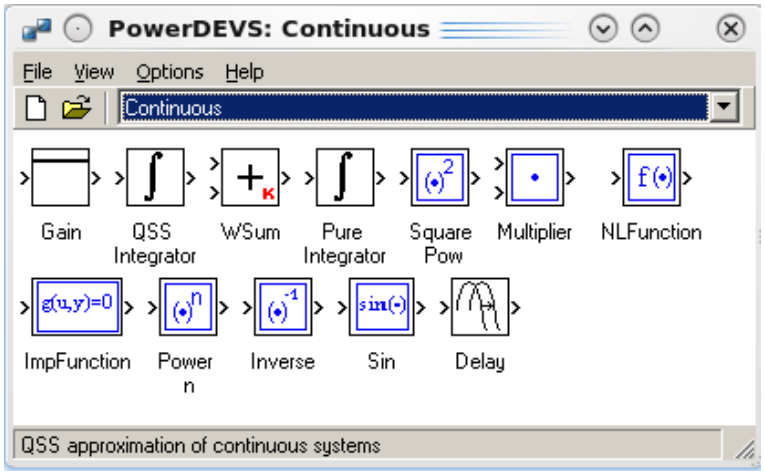


Figura 2.8: Interfaz gráfica de usuario principal de PowerDEVS.

PowerDEVS puede intercambiar parámetros con Scilab durante la simulación del mismo modo que lo hacen Simulink y Matlab. Esta interacción permite explotar todas las capacidades de procesamiento de datos, visualización, tratamiento matemático y manipulación de matrices de Scilab, dándole a su vez al usuario un espacio de trabajo interactivo.

A pesar de que PowerDEVS fue diseñado originalmente para funcionar en plataformas Windows, se desarrolló una versión ad–hoc de Kubuntu 8.04 que incluye PowerDEVS y módulos de Linux de tiempo real (RTAI). El motor de PowerDEVS incluye módulos que usan funciones RTAI para sincronizarse con el tiempo real, manejar interrupciones, medir el tiempo de CPU, etc. Los módulos de tiempo real permite a un modelo atómico DEVS sincronizar la ocurrencia de un evento con el reloj de tiempo real con una precisión de aproximadamente 2 μsec y medir el tiempo físico con una precisión del orden de los nanosegundos [BKBZ08]. De esta manera, el usuario puede realizar simulaciones de tiempo real

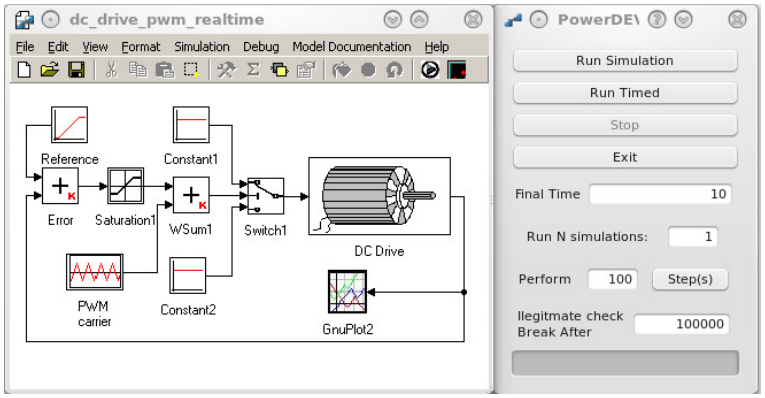


Figura 2.9: Modelo de un motor con control de velocidad en PowerDEVS.

usando los métodos QSS de una manera muy conveniente. PowerDEVS/RTAI resulta más fácil de utilizar que el difundido Real Time Workshop de Matlab/-Simulink, ya que PowerDEVS permite modelar y simular sin cambiar de sistema operativo.

PowerDEVS es una herramienta pensada para ejecutar en procesadores de propósito genérico, mayormente en entornos tipo PC. Sin embargo, la velocidad de procesamiento en tiempo real de las redes de datos de alta capacidad requieren de procesadores de propósito específico. Esto limita, en principio, la portabilidad de los modelos de control de red diseñados con PowerDEVS. Por ello, para la etapa de implementación de modelos en aplicaciones reales, se recurrirá a otra herramienta basada en DEVS, apta para ejecutar embebida en procesadores de propósito específico. Dicha herramienta es Embedded CD++, y se describe en la sección siguiente.

2.4.2. Embedded CD++ (ECD++)

CD++ [Wai02] es una herramienta de simulación de código abierto que implementa el formalismo de simulación DEVS. Al igual que PowerDEVS, en CD++ los simuladores y coordinadores avanzan la simulación intercambiando mensajes del modo especificado por el mecanismo de simulación abstracto de DEVS. CD++ está diseñado siguiendo el paradigma de orientación a objetos, permitiendo al usuario integrar de forma robusta y sencilla las bibliotecas de simulación con cada nuevo código de modelado necesario para representar un sistema bajo estudio.

Un sistema en tiempo real es aquel que provee un comportamiento cuya correctitud depende tanto de los datos resultantes como del tiempo en el cual dichos resultados son producidos [Sta96]. Cuando un sistema produce un resultado correcto pero en un instante posterior al límite temporal permitido (deadline), el resultado podría considerarse incorrecto. En este contexto, un simulador de tiempo real debe tener la capacidad de asociar la producción de eventos con el tiempo físico, y de controlar si los deadlines son cumplidos o no.

RT-CD++ [WGM05] es una extensión de la herramienta CD++ que permite la simulación en tiempo real. Reemplaza las funciones de manejo de tiempo virtual por funciones de avance de tiempo sincronizadas con el tiempo físico

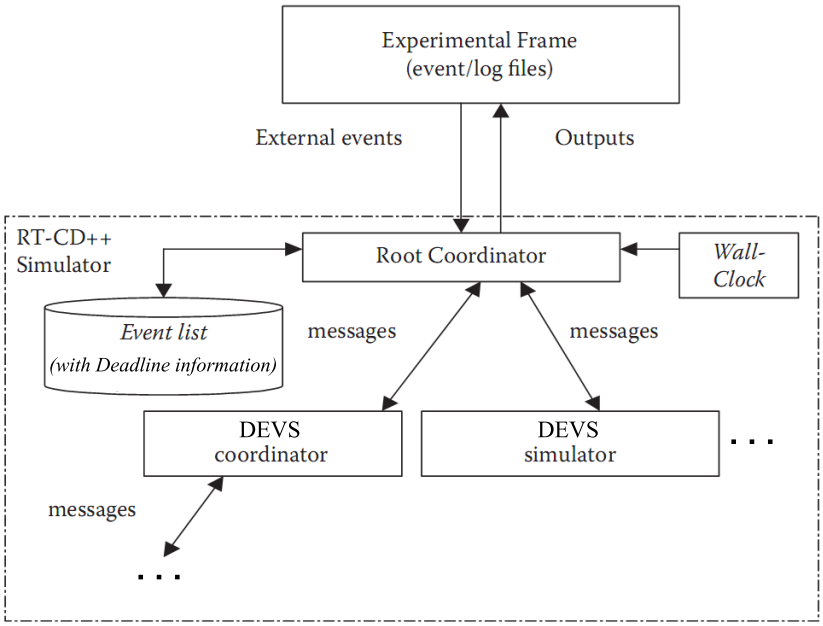


Figura 2.10: Esquema de simulación de ECD++

(provisto por el reloj de la plataforma computacional). La Figura 2.10 describe la jerarquía de software generada para ejecutar un sistema definido por modelos DEVS atómicos y acoplados. El coordinador raíz (Root Coordinator) administra la interacción con el marco experimental (Experimental Frame) adoptado para realizar ensayos con el modelo, y reporta salidas (Outputs) para el análisis de resultados.

Cada componente coordinador (DEVS Coordinator) sincroniza los subcomponentes que debe administrar (uno o varios DEVS Simulator y/o DEVS Coordinator) acorde a la jerarquía definida. Los componentes Coordinator implementan las definiciones estructurales (modelos acoplados DEVS) mientras que los componentes Simulator implementan las definiciones comportamentales (modelos atómicos DEVS).

Root Coordinator puede verse como un caso especial de Coordinator, siendo el encargado de administrar, controlar y hacer avanzar el tiempo global de ejecución. También recibe eventos externos y los transforma en mensajes aptos para ser manipulados por los modelos atómicos y/o acoplados subyacentes.

Las restricciones de tiempo real (deadlines) pueden especificarse y asociarse a los mensajes externos, de modo tal que cuando un mensaje avanza en su procesamiento los Coordinators pueden verificar si el deadline fue violado o no, y tomar decisiones en base a ello.

RT-CD++ ha sido ensayado extensivamente utilizando tanto aplicaciones reales como benchmarks sintéticos. En todos los casos se reportaron overheads pequeños (del orden del 2 % a 3 % para modelos razonablemente grandes). Esto se debe en gran parte a la definición de un coordinador principal de jerarquía achatada (Flat Coordinator) que mejora el desempeño del simulador gracias a

la disminución de la cantidad de mensajería interna.

El Simulador CD++ Embebido (ECD++, por *Embedded CD++ Simulator*) [YW07] es una adaptación de RT-CD++ para poder ejecutar en entornos embebidos (con recursos restringidos) e interactuar con dispositivos de hardware externos. Adicionalmente ECD++ soporta la simulación de modelos especificados con el paradigma extendido P-DEVS (Parallel DEVS), que permite el tratamiento explícito de múltiples eventos simultáneos y la activación simultánea de múltiples modelos atómicos. La forma estándar de desarrollar para ECD++ es mediante la edición de código C++ con editores de texto. Sin embargo se han desarrollado entornos avanzados para diseño visual que forman parte del trabajo realizado en esta Tesis, y serán descritos en el Capítulo 5.

2.5. DEVS y Modelado de redes de datos

Las redes digitales de comunicaciones presentan ciertas dificultades para su modelado y simulación, las cuales suelen agravarse cuando su tamaño crece.

Las representaciones más detalladas (a nivel de paquetes individuales) consisten en modelos de eventos discretos capaces de capturar comportamientos exactos (enrutamiento, pérdida, retransmisión, retardos, etc.) Estos modelos [IH08, OMN04] presentan la desventaja de tener un costo computacional muy elevado para simular topologías de red complejas y/o intensidades de tráfico considerables. Tampoco son útiles para realizar análisis de propiedades globales (por ejemplo, sensibilidad de performance ante variaciones de parámetros).

Las representaciones menos detalladas (a nivel de flujos de paquetes) consisten en modelos continuos que aproximan la dinámica de la red mediante variables promediadas (longitud de colas, tasas de transmisión, tasas de pérdida). Estos modelos [MGT99, MGT00, FHPW00] permiten resolver en tiempos muy bajos sistemas de alta complejidad y por su formulación matemática permiten aplicar técnicas analíticas para obtener conclusiones generalizadas. Sin embargo, presentan la desventaja de capturar sólo dinámicas lentas o de régimen estacionario, ya que las variables son promediadas sobre períodos relativamente grandes de tiempo.

Una alternativa integradora y de gran potencial es aplicar técnicas híbridas de modelado y simulación [KM01, BHLO03, GLT04, YS07]. Sin embargo, todas las propuestas que ofrece la bibliografía evidencian la necesidad de realizar al menos un cambio de formalismo y/o herramienta.

Surge entonces de modo natural recurrir al formalismo DEVS como marco integrador, tanto formal como práctico, para el modelado híbrido de redes de datos. Conceptualmente, por su capacidad inherente para representar sistemas discretos, permite modelar de modo exacto cualquier tipo de protocolo de comunicaciones. A su vez, haciendo uso de las extensiones de DEVS para representación de sistemas continuos (Sección 2.3) permite incorporar aproximaciones continuas de las redes, haciendo coexistir los modelos discretos y continuos sin cambiar de formalismo ni de herramienta. Por último, cuando se pretende utilizar una parte del sistema diseñado para implementarse en una aplicación real, existen herramientas que permiten transformar los modelos DEVS originales en piezas de software ejecutando en tiempo real (Sección 2.4.2) evitando la necesidad de reescritura de la lógica de los modelos.

Siguiendo esta idea, se planteará un escenario de estudio motivador y se

analizará la viabilidad de aplicar DEVS como solución integradora. Se verá que existen limitaciones de orden teórico y práctico, y se especificarán las necesidades a cubrir para intentar superarlas.

2.5.1. Escenario de Estudio Motivador: Control de Congestión en Internet

Las redes de paquetes de datos de propósito genérico como Internet se caracterizan por tener una topología dinámica, una demanda de tráfico impredecible y requerimientos de calidad de servicio heterogéneos, debido a la diversidad de aplicaciones que la utilizan (multimedia, transferencia de archivos, transacciones comerciales, navegación, etc.). Por su característica de acceso abierto e irrestricto a una cantidad de usuarios desconocida, se producen frecuentemente situaciones perjudiciales de saturación y sub-utilización de recursos. La consecuencia es la degradación de la calidad de servicio por una pérdida global de performance. Esto se produce en general por la conjunción de dos factores: una tasa elevada de pérdida de paquetes (por saturación de buffers finitos en los nodos enrutadores) y una subutilización del ancho de banda disponible (en el medio físico), disminuyendo sensiblemente la calidad de servicio de la red. Este fenómeno puede inestabilizarse y entrar en un ciclo de realimentación positiva conocido como *colapso de congestión* que conduce a la inoperatividad total de la red [Nag84].

Para evitar esta situación, se diseñan protocolos de comunicación [Jai89, MS90, SA91, WC91, WC92] con estrategias dirigidas a controlar la calidad de servicio, basándose en mecanismos de señalización que permiten mantener un conocimiento actualizado del estado de la red. El objetivo es maximizar la tasa de transmisión minimizando los retardos, la tasa de pérdida de información, y la tasa de retransmisión por errores.

Una característica de estas estrategias es que la propia señalización de control se ve afectada por los síntomas de la congestión que se quiere controlar, experimentando retardos variables impuestos por la latencia de vínculo, el ancho de banda, la tasa de pérdida de paquetes, los retardos de encolamiento y los cambios de topología. Por ello, se los considera *sistemas de control distribuido con retardos variables* [HNX07], existiendo un creciente interés en obtener modelos que permitan estudiar sus propiedades en términos de eficiencia, robustez y estabilidad [MN05].

Control Supervisorio, Control de Admisión y Control de Congestión con TCP

Una aplicación de gran interés académico y práctico consiste en modelar el Protocolo de Control de Transporte (TCP, por *Transport Control Protocol*) que controla aproximadamente el 90 % del tráfico de datos de Internet [Mas05] operando en la Capa 3 del modelo de referencia OSI [Zim02].

En la Figura 2.11 se muestra un escenario típico para TCP, en el cual se observa que a nivel local (en cada extremo de una comunicación) forma parte de un conjunto de protocolos, y que a nivel global (incluyendo los extremos y la red intermedia) forma parte de un sistema de control de congestión. Por simplicidad, se considera a la red como un único elemento enrutador intermedio, ya que aporta una complejidad suficiente para estudiar un control de congestión.

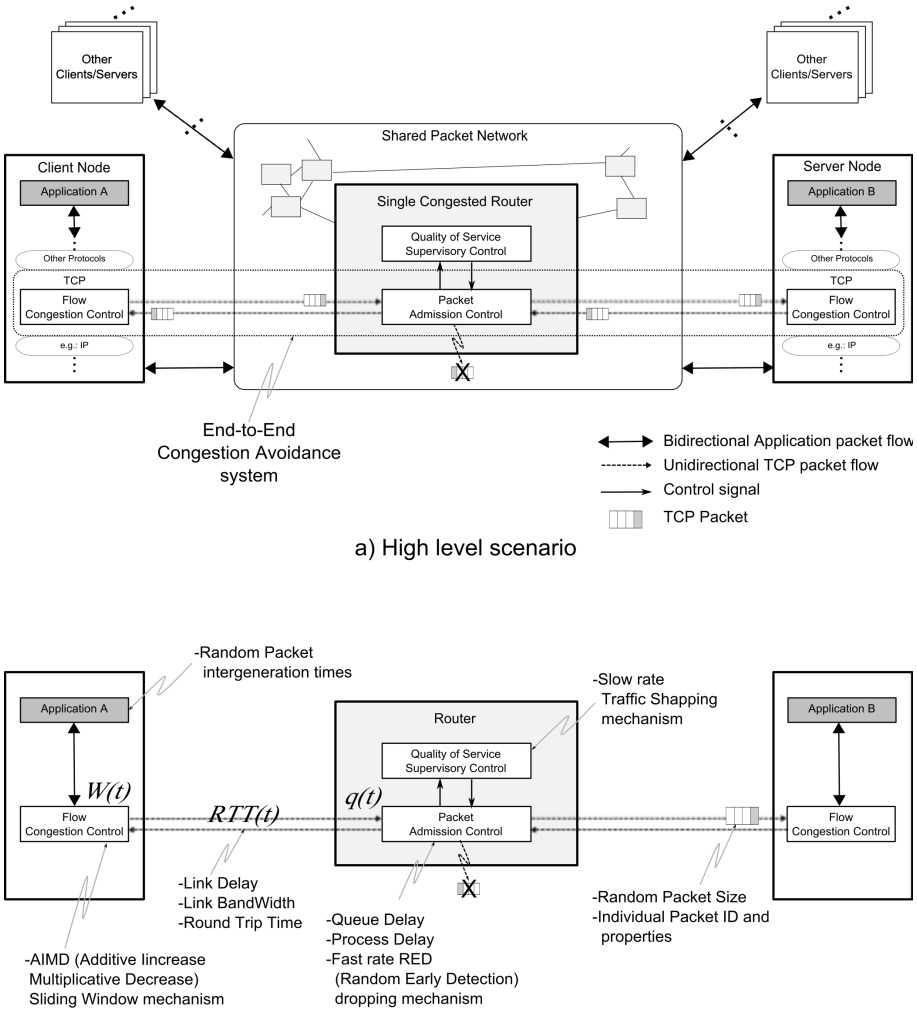


Figura 2.11: Escenario de estudio motivador.

La función básica de TCP es garantizar para cada conexión entre dos nodos la entrega ordenada de paquetes enviados sobre la red, detectando las posibles pérdidas de información y efectuando retransmisiones si es necesario. Entre muchas otras funciones que cumple TCP (establecimiento, administración y finalización de cada conexión) cuenta con un Módulo de Control de Congestión. Dicho módulo implementa un algoritmo que opera simultánea y coordinadamente en el extremo emisor y receptor, ignorando en principio la complejidad de la red subyacente. Por ello, se lo considera un algoritmo de control distribuido.

En Internet cada conexión TCP comparte enrutadores y enlaces, y compite con una cantidad desconocida de flujos de paquetes (controlados por TCP o no). En este contexto el Control de Congestión de TCP busca sacar el mayor provecho del ancho de banda disponible, pero evitando a su vez provocar un colapso

de congestión. Es decir, se busca minimizar la contribución de una conexión a la generación de congestión en la red compartida, pero *intentando a la vez* maximizar la utilización del ancho de banda disponible.

El primer algoritmo de control de congestión para Internet fue incorporado en TCP a partir del trabajo de [JK88]. Allí se definió que la señalización para que un extremo emisor tome conocimiento del estado de congestión de la red, sería por medio de paquetes especiales notificando el éxito o fracaso de la correcta recepción en el extremo receptor. Cuando los buffers de los enrutadores intermedios se saturan, las colas alcanzan su tamaño máximo $q(t) = q_{max}$ y los nuevos paquetes entrantes se descartarán, disparando así una señalización de congestión. Este es un mecanismo básico de Control de Admisión.

Desafortunadamente, debido al crecimiento de la capacidad de las redes, surge el problema del crecimiento acorde de la longitud de cola en los enrutadores provocando el incremento de los retardos y con ello la profundización de la congestión.

Como consecuencia, debieron incorporarse algoritmos de Control de Admisión más sofisticados en los enrutadores. La idea mejorada es evitar que la señalización de pérdida de paquetes ocurra abruptamente una vez que las colas se saturan, sino que en cambio ocurra gradualmente, al estilo de una "señalización temprana". Se consigue así una *degradación suave* que permite alojar ráfagas esporádicas de paquetes (saturando momentáneamente las colas) pero manteniendo el *promedio de ocupación* en valores moderados hasta que la señalización temprana surta efecto.

Estos algoritmos pertenecen a la categoría Administración Activa de Colas (AQM, por *Active Queue Management*), y si bien existen numerosas propuestas [LdF08, ALLY02] los de mayor interés se diseñan para operar en cooperación con el control de TCP, lo que da lugar a técnicas combinadas denominadas TCP/AQM. TCP cumple su parte implementando un mecanismo llamado Prevención de Congestión de TCP (CA/TCP, por *TCP Congestion Avoidance*) cuyo objetivo es reaccionar tempranamente ante una señal de congestión incipiente y limitar su tasa de transmisión hasta que la congestión desaparezca.

El principio de regulación de Prevención de Congestión de TCP (CA/TCP) utiliza un sistema de *ventana deslizante* [CHdV03] por medio del cual, de todos los paquetes disponibles para transmitir, serán enviados a la red sólo una cantidad $W(t)$ limitada. Dicho tamaño de ventana se controla haciéndola crecer en caso de no existir indicación de congestión, o disminuyéndola en caso contrario, controlando así la tasa de tráfico ofrecida a la red. La actualización del tamaño de la ventana se produce tanto ante una llegada de un mensaje de aceptación exitosa de paquetes por parte del extremo receptor (paquete de Acknowledge - ACK), como ante una indicación de pérdida de paquete. CA/TCP garantiza que nunca existirán en la red más de W paquetes sin aceptación exitosa (o paquetes *"en vuelo"*). Tanto la ausencia de un *ACK correcto* (en un lapso tolerable de tiempo), como la llegada de un *ACK incorrecto* (en cualquier instante) es interpretada como señal de congestión.

La dinámica de CA/TCP es dominada por el Tiempo de Ida y Vuelta (RTT, por *Round Trip Time*). $RTT(t)$ es el tiempo que tarda el sistema en detectar una condición de congestión en $q(t)$, producir la señalización hasta el emisor, reaccionar modificando W(t), y finalmente influir en la condición de $q(t)$.

Por último, entra en juego la necesidad de adaptabilidad ante escenarios cambiantes. Un conjunto satisfactorio de parámetros de operación para TCP/AQM

puede tener validez para ciertas condiciones del estado de tráfico y para cierta política de calidad de servicio ofrecida por un proveedor de red. En la práctica, ambas situaciones pueden cambiar en el lapso de minutos, horas o días. Evidentemente es imposible diseñar protocolos que vayan a reemplazarse o reconfigurarse manualmente en tiempos tan breves. Para ello, se diseñan sistemas de control de dinámica lenta conocidos como Control Supervisorio de Calidad de Servicio. Por un lado, brindan la posibilidad de monitorear el estado de la red en tiempo real, y acorde a ciertas reglas de control, modificar los parámetros del Control de Admisión supervisado. Esto permite expandir el rango de operación de las estrategias de CA/TCP. Por otro lado, cuando se requiera modificar las políticas de Calidad de Servicio, simplemente se actualizan las reglas de control y se obtiene una nueva dinámica de la red con tiempo nulo de interrupción del servicio. Un caso de aplicación muy interesante es poder modificar en tiempo real el umbral de nivel de llenado de buffer en un enrutador a partir del cual Administración Activa de Colas (AQM) comenzará a descartar paquetes en forma temprana.

Con esto, el escenario de estudio presenta 3 sistemas de control operando en simultáneo y de manera cooperativa para administrar la congestion en una red TCP.

2.5.2. Necesidades y Limitaciones

La primer necesidad de modelado que se evidencia en el escenario de estudio es la de representar aleatoriedad. Una conexión TCP se ve sometida simultáneamente a varios fenómenos estocásticos que afectan su dinámica (ver Figura 2.11), siendo los más importantes: la tasa de tráfico generado en un nodo emisor (demanda hacia la red), la distribución probabilística del tamaño de paquete, y la tasa de descarte de paquetes. La aleatoriedad en $RTT(t)$ es esencialmente una función de los fenómenos anteriores.

Desafortunadamente, la definición clásica de DEVS [ZKP00] no permite representar sistemas estocásticos. Esto es así ya que $\delta_{\text{int}}(\cdot)$,$\delta_{\text{ext}}(\cdot)$,$\lambda(\cdot)$ y $t_a(\cdot)$ fueron definidas como funciones deterministas, es decir, asignan un valor único a cada valor posible de sus argumentos.

Esto representa una evidente limitación formal. La práctica usual adoptada por la comunidad DEVS frente esta limitación ha sido incluir variables pseudoaleatorias en la definición formal del modelo. Por ejemplo, se define una función de transición interna como $\delta_{\text{int}}(\cdot, r)$ en donde r es el resultado de tomar un número pseudoaleatorio distribuido uniformemente provisto por el generador de secuencias pseudoaleatorias de algún lenguaje de programación. A partir de r puede obtenerse cualquier distribución de probabilidades de interés práctico, dotando así a δ_{int} de las propiedades estocásticas deseadas. Pero este enfoque incluye una trampa, ya que es que en rigor $\delta_{\text{int}}(\cdot, r)$ modela una función determinista por ser r una secuencia discreta de números conocida a priori.

Para permitir que DEVS especifique de modo riguroso procesos estocásticos es necesario redefinir el formalismo para brindarle la capacidad de manipular espacios de probabilidad. En el Capítulo 3 de esta Tesis se propone una solución, consistente en el nuevo formalismo STDEVS.

Otra necesidad es la de estudiar la dinámica de TCP/AQM con aproximaciones continuas. En la literatura se ha propuesto una variedad de modelos basados en ecuaciones diferenciales no lineales para estudiar la dinámica de TCP

[PFTK98, HMTG01]. Los modelos continuos posibilitan el estudio analítico de las propiedades matemáticas de los controles involucrados. Asimismo, independientemente de la eficiencia del simulador que se elija, existe una limitación intrínseca de escalabilidad de los modelos discretos para efectuar simulaciones de centenas de flujos de paquetes TCP, a tasas de transmisión de varios megabits por segundo. En cambio, utilizando aproximaciones fluidas de los flujos de paquetes y sus algoritmos de control se pueden estudiar problemas de magnitud importante en un tiempo de simulación razonable con una capacidad de cómputo modesta.

Como ya se ha dicho DEVS provee la capacidad de aproximar sistemas continuos con tanta precisión como se desee. Sin embargo, no existe una metodología que establezca pasos ordenados para poder cumplir un ciclo de vida completo de un modelo: desde su primer formulación matemática como sistema continuo hasta su implementación final como pieza de software ejecutando en tiempo real, pasando por etapas intermedias de verificación que refinen incrementalmente los modelos con sus equivalentes a tiempo discreto y/o eventos discretos (según convenga).

Esto puede verse como una limitación metodológica, que atenta contra la adopción de DEVS como tecnología unificadora para el control de redes de datos. En el Capítulo 6 de esta Tesis se propone una metodología punta a punta, aplicada a un caso de estudio de Control de Admisión, en donde los modelos continuos son utilizados inicialmente tanto para representar la red como para diseñar el módulo de control aplicando Teoría de Control clásica (continua y discreta). El Control de Admisión se estudia como un control local, ubicado en el enrutador de la Figura 2.11.

Sin embargo, cuando se desea aplicar modelado continuo al Control de Congestión conformado por el Control de Flujo de TCP más el Control de Admisión del enrutador (es decir, cuando se trata con el sistema distribuido TCP/AQM) surgen nuevas complicaciones. Los modelos continuos que representan a TCP resultan ser ecuaciones diferenciales de la familia de las Ecuaciones Diferenciales con Retardo (DDE). En estos sistemas las variables de estado del modelo dependen tanto de las variables del sistema en el instante actual como de sus evoluciones en el pasado. Concretamente, el tamaño $W(t)$ de la ventana tendrá una descripción del tipo $\dot{W}(t) = f(\cdot, W(t - RTT(t)))$ en donde la derivada de W en cada instante depende, entre otras cosas, del valor que W asumió RTT unidades de tiempo atrás. Aún más, el tiempo RTT es también variable en el tiempo, con $RTT(t) = RTT(\cdot, q(t))$, indicando que depende, entre otras cosas, de la longitud de la cola en el enrutador. Esta familia de modelos excede a TCP/AQM siendo de suma importancia para diseñar cualquier tipo de control distribuido sobre redes, ya que las señales de sensado y actuación siempre están sometidas a las condiciones de retardo impuestas por la red.

Desafortunadamente, se encuentra una nueva limitación, dado que los métodos de integración QSS basados en cuantificación de estados no tienen la capacidad de manipular información histórica de las variables continuas. En el Capítulo 4 de esta Tesis se propone una nueva clase de algoritmos de integración denominada DQSS para resolver numéricamente ecuaciones diferenciales con retardos, basados en técnicas de cuantificación de estados en lugar de la técnica clásica de discretización del tiempo. Para ello, se reformulará la representación genérica de ODE en Ec. (2.1) para incorporar información retardada y luego se aplicará una cuantificación a las variables de estado.

Finalmente, con las nuevas herramientas aportadas están dadas las condiciones para estudiar de manera integral, eficiente y práctica el Control de Congestion del escenario de estudio utilizando DEVS. En el Capítulo 7 se presenta el estudio conjunto de un Control de Flujo tipo AIMD (Additive Increase Multiplicative Decrease) implementado por TCP, con un Control de Admisión tipo Detección Temprana Aleatoria (RED, por *Random Early Detection*) implementado por un enrutador.

En primer lugar, se estudia el sistema por medio de su aproximación fluida, haciendo uso de los nuevos métodos de integración DQSS.

En segundo lugar se estudia el mismo sistema pero completamente discreto, haciendo uso de la nueva teoría STDEVS y las nuevas estructuras de paquetes. Al comparar los tiempos de simulación, se confirma la necesidad planteada más arriba acerca de contar con aproximaciones fluidas de los modelos para ganar escalabiblidad. Efectivamente, con el modelado continuo se logra disminuir los tiempos de manera sensible, manteniendo una representatividad cualitativamente satisfactoria.

En tercer y último lugar, se implementa una estrategia de modelado de flujos híbridos combinando la ventaja de la eficiencia computacional de una simulación fluída con la ventaja del detalle granular provisto por una simulación discreta. Por medio de DEVS y las nuevas herramientas teórico–prácticas presentadas en esta Tesis, se podrán modelar y simular flujos híbridos bajo un formalismo matemático unificado y en una misma herramienta; en contraste con los antecedentes de los que da cuenta la literatura basados en modificaciones ad–hoc de motores de simulación específicos.

Capítulo 3

DEVS y Sistemas Estocásticos

El presente capítulo intentará ofrecer una solución para la primer limitación identificada en la Sección 2.5.2, que consiste en la incapacidad formal de DEVS para representar fenómenos estocásticos.

Esta es una limitación que afecta al modelado de cualquier aplicación con DEVS. En particular, en el contexto del escenario de estudio motivador, es de interés poder modelar la aleatoriedad presente en una conexión TCP afectando su dinámica (por ejemplo, la tasa de tráfico generado en un nodo emisor, la distribución probabilística del tamaño de paquete, la tasa de descarte de paquetes, etc.)

En este capítulo se presenta una extensión a la definición original del formalismo DEVS para incluir formalmente propiedades estocásticas. Recurriendo al uso de la teoría de Espacios de Probabilidad, se define la especificación de Sistemas de Eventos Discretos Estocásticos (STDEVS, por *STochastic Discrete EVent Systems Specification*), la cual provee un marco formal para modelado y simulación de sistemas de eventos discretos *genéricos no–deterministas*. Se estudiarán las principales propiedades teóricas del formalismo STDEVS, incluyendo una nueva definición de *legitimidad* de modelos en un contexto estocástico y una nueva prueba para la *clausura bajo acoplamiento*. Asimismo se ilustrarán las nuevas capacidades de *modelado* provistas por STDEVS y su relación con aquellas provistas por DEVS clásico.

Se proveerán ejemplos prácticos de simulación involucrando el análisis de performance de un sistema de cómputo y el modelado híbrido de un sistema de control conectado en red, es decir, aplicaciones donde el modelado de componentes estocásticos es *vital*. La aplicación de la nueva herramienta STDEVS a un caso práctico de control de congestión de red como el planteado en el escenario motivador será realizada en la Sección 7.3.2.

3.1. Introducción

A pesar de que las principales herramientas desarrolladas en los últimos años [ZS00, WCD01, FDB02, KLP03] para modelar y simular modelos DEVS han incorporado el uso de funciones aleatorias, la *definición teórica formal* de DEVS

considera únicamente sistemas deterministas. Esta característica limita seriamente la extensión del formalismo a una vasta familia de sistemas estocásticos.

En el presente capítulo se presenta un nuevo formalismo basado en DEVS para modelar sistemas de eventos discretos estocásticos generalizados, al cual se denominará STDEVS. Las principales propiedades teóricas de STDEVS (clausura bajo acoplamiento, legitimidad, etc.) se redefinirán formalmente en el nuevo contexto de sistemas estocásticos.

Los modelos estocásticos juegan un rol preponderante dentro de la Teoría de Sistemas de eventos discretos. De hecho, cualquier sistema involucrando incertidumbres, acciones impredecibles de humanos, o fallas no planificadas requiere un tratamiento no determinista. Algunos ejemplo de formalismos tradicionales de eventos discretos estocásticos son las Cadenas de Markov [Cut80], Redes de Colas [Cas93] y Redes de Petri Estocásticas [AMBC+95]. Estas técnicas permiten analizar y simular modelos estocásticos en diversas aplicaciones. Existen algunos antecedentes de trabajos que trataron el vínculo entre DEVS y sistemas estocásticos [Agg75, Mel76, Jos96]. Sin embargo, ninguno ha provisto una teoría general ni un soporte teórico formal para modelos estocásticos DEVS.

La metodología basada en STDEVS, además de proveer un mecanismo de especificación genérico, describe el comportamiento probabilístico a nivel de *especificación de transición de estados*, y cubre las propiedades de acoplamiento de manera directa. STDEVS hereda de DEVS las capacidades de multimodelado para representar sistemas híbridos y las ventajas de performance para su simulación en comparación con técnicas tradicionales, cuando se combinan sistemas estocásticos de tiempo discreto y tiempo continuo.

El capítulo está organizado del siguiente modo. En la Sección 3.2 se provee un repaso de intentos previos de relacionar DEVS y fenómenos estocásticos, y la teoría de Espacios de Probabilidad (que utilizaremos para definir STDEVS en términos de conjuntos genéricos de estados). La Sección 3.3 presenta varios problemas de modelado a modo de motivación y formula ejemplos para hacer evidente las principales limitaciones del formalismo DEVS clásico para representar sistemas estocásticos genéricos. La Sección 3.4 describe la idea principal que hace evidente la necesidad de proponer STDEVS y define formalmente a STDEVS. Luego la Sección 3.5 muestra que STDEVS es cerrado bajo acoplamiento y define la propiedad de legitimidad. En la Section 3.6 se muestra que cualquier modelo *DEVS medible* cuyas funciones de transición dependen de variables aleatorias, define un modelo STDEVS equivalente, permitiendo modelar ciertos modelos STDEVS sin necesidad de utilizar espacios de probabilidad. Esto último opera como marco formal para aquellas herramientas de simulación que utilizan generadores de secuencias pseudo–aleatorias. Inmediatamente se provee un resumen y discusión para sustentar el beneficio de STDEVS por medio de la resolución de los problemas planteados en la Sección 3.3. Finalmente el presente Capítulo termina con la Sección 3.7 que ilustra la aplicación de STDEVS en el contexto de un *proceso de modelado estocástico* por medio de distintos ejemplos aplicados a computadoras y redes involucrando modelado híbrido (continuo/discreto) y Teoría de Control.

3.2. Antecedentes

3.2.1. Relaciones previas entre DEVS y fenómenos estocásticos

Desde los comienzos del formalismo DEVS se hizo evidente que existía la necesidad de establecer una relación formal que le permita describir las características estocásticas de los sistemas estudiados. En [Agg75, Zei76] los autores muestran que las simulaciones de eventos discretos que avanzan acorde a secuencias pseudo–aleatorias definen un modelo DEVS equivalente. En otras palabras, el formalismo DEVS puede capturar el comportamiento de sistemas estocásticos que son simulados utilizando secuencias pseudo–aleatorias, es decir, que ejecutan un *modelo determinista de un sistema estocástico* [Zei76]. Sin embargo, lo antedicho no provee una metodología para describir modelos DEVS estocásticos, en los cuales la aleatoriedad se encuentre a nivel de la especificación formal del sistema, y no del mecanismo de simulación. En [Mel76] el autor establece una relación entre resultados de experimentos aleatorios y la evolución temporal de una simulación DEVS, al asignar medidas de probabilidad a las trayectorias de estados que son observables a nivel de entradas y salidas. Sin embargo, este trabajo está limitado a modelos descritos a nivel de trayectorias externas de entrada/salida [ZKP00] (es decir, no es capaz de especificar la dinámica de los modelos a nivel de transiciones de estados). El primer enfoque verdaderamente estructural para tomar en cuenta el comportamiento estocástico interno a nivel de transiciones de estados fue bosquejado en [Jos96]. Sin embargo, dicho trabajo se limitó a tratar modelos DEVS con conjuntos de estados finitos. Dado que una de las propiedades más importantes de DEVS es su capacidad de manejar conjuntos de estados arbitrariamente genéricos, la restricción a conjuntos finitos de estados es un problema. Los trabajos mencionados redundaron en un conjunto de resultados útiles que han sido suficientes para enfrentar algunos dominios de problema puntuales, pero únicamente desde una perspectiva comportamental, no estructural ni genérica.

3.2.2. Espacios de Probabilidad

Como se verá, para superar las dificultades expuestas en los párrafos precedentes, se definirá STDEVS en términos de espacios genéricos de probabilidad, descansando en la teoría de Espacios de Probabilidad [GD04, Bil95]. Un *espacio muestral* S_{sp} de un experimento aleatorio es un conjunto que incluye todos los posibles resultados del experimento.

Una *sigma–álgebra* [1] $\mathcal{F}$ del espacio muestral S_{sp} es una colección no vacía de subconjuntos de elementos tomados de S_{sp}. La idea es que cuando se trabaja con un espacio muestral continuo S_{sp}, es posible asignar probabilidades a subconjuntos de S_{sp} y no a elementos individuales de S_{sp}. Luego, las probabilidades se asignan a elementos de la sigma–álgebra $\mathcal{F}$ (es decir, a subconjuntos de S_{sp}). Una sigma–álgebra no puede ser una colección arbitraria de subconjuntos de S_{sp}. Una colección $\mathcal{F}$ debe satisfacer las siguientes propiedades para constituir un sigma–álgebra:

[1] Una sigma–álgebra es denominada a veces un *espacio de eventos*, pero se evitará utilizar este término en esta Tesis para evitar la confusión con el significado que se le asigna a la palabra *evento* en el contexto de la metodología DEVS.

- si $F \in \mathcal{F}$ entonces $F^c \in \mathcal{F}$ (donde F^c es el complemento de F en S_{sp}, $F^c \cup F = S_{sp}$).
- si $F_i \in \mathcal{F}$ para $i = 1, \ldots, \infty$, entonces también $\bigcup_{i=1}^{\infty} F_i \in \mathcal{F}$.

Notar que dado que $F^c \cup F = S_{sp}$, las últimas dos condiciones implican que $S_{sp} \in \mathcal{F}$ y también que $\oslash \in \mathcal{F}$. En la Figura 3.1 se muestra una diagrama representando una sigma–álgebra generada desde un espacio muestral arbitrario y sus miembros.

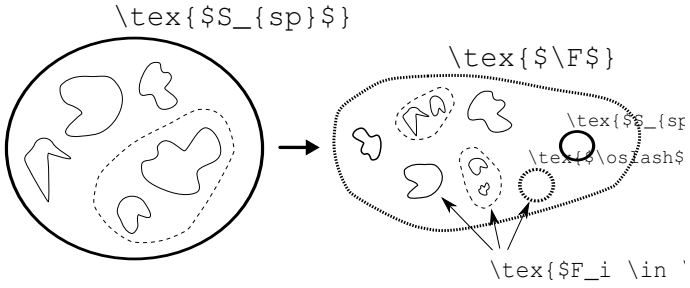


Figura 3.1: Un espacio muestral S_{sp} y sus sigma–álgebra generadas $\mathcal{F}$. Cada miembro F_i de la colección $\mathcal{F}$ satisface las condiciones requeridas impuestas a constituir una sigma–álgebra. El espacio vacío $\oslash$ y el espacio muestral original S_{sp} son miembros requeridos de $\mathcal{F}$.

Estas condiciones son necesarias para construir una estructura genérica y medible partiendo desde S_{sp}, el cual puede estar equipado con medidas de probabilidad como se verá en breve.

Un ejemplo particular de una sigma–álgebra sobre un espacio muestral S_{sp} es la colección de todos los posibles subconjuntos de S_{sp} ($2^{S_{sp}}$, denominado el *conjunto potencia* de S_{sp}).

A pesar de que este no es una sigma–álgebra (ya que puede incluir muchos subconjuntos sin interés, a los cuales no se desea –o incluso no es posible– asignarles medidas de probabilidad), es útil en varias pruebas teóricas.

En la mayoría de los problemas prácticos, se deberían describir únicamente las probabilidades de *ciertos subconjuntos* de S_{sp} (p.ej., si S_{sp} fuese el conjunto de los reales $\Re$, normalmente se asignarían probabilidades solo a intervalos abiertos). Si bien una colección de estos subconjuntos elegida arbitrariamente (que puede ser de interés para trabajar en un problema particular) puede no constituir una sigma–álgebra en si misma, *genera siempre* una sigma–álgebra. Tómese a $\mathcal{G}$ como una colección particular de subconjuntos de S_{sp}. La sigma–álgebra generada por $\mathcal{G}$, denotada $\mathcal{M}(\mathcal{G})$, es la ínfima sigma–álgebra que contiene todos los elementos de $\mathcal{G}$.

Un par $(S_{sp}, \mathcal{F})$ consistiendo en un espacio muestral S_{sp} y un campo sigma $\mathcal{F}$ de subconjuntos S_{sp} es denominado un *espacio medible*. Una *medida de probabilidad* P sobre un espacio medible $(S_{sp}, \mathcal{F})$ es una asignación de un número real $P(F)$ a cada miembro F de un campo sigma, de tal modo que P obedece las siguientes reglas:

- Axioma 1. $P(F) \geq 0$ para todo $F \in \mathcal{F}$ (las probabilidades son no negativas).

- Axioma 2. $P(S_{sp}) = 1$ (la probabilidad de un resultado sobre el espacio muestral completo es igual a 1).

- Axioma 3. Si $F_i \in \mathcal{F}$, $i = 1, \ldots, \infty$ son conjuntos disjuntos, luego $P(\bigcup_{i=1}^{\infty} F_i) = \sum_{i=1}^{\infty} P(F_i)$.

Cuando un campo sigma $\mathcal{F} = \mathcal{M}(\mathcal{G})$ generado desde una colección particular de interés práctico $\mathcal{G}$ de subconjuntos de S_{sp}, el conocimiento de $P(G)$ para cada subconjunto $G \in \mathcal{G}$ inmediatamente define la función P para cada subconjunto $F \in \mathcal{F}$.

Finalmente, el experimento aleatorio puede ser descrito completamente por un *espacio de probabilidad* definido por el triplete $(S_{sp}, \mathcal{F}, P)$ consistiendo de un espacio muestral S_{sp}, una sigma–álgebra $\mathcal{F}$ de subconjuntos de S_{sp}, y una medida de probabilidad P definida por todos los miembros de $\mathcal{F}$.

Sintetizando, para cada $F \in \mathcal{F}$, $P(F)$ expresa la probabilidad de que un experimento aleatorio de como resultado una muestra $s \in F \subseteq S_{sp}$.

3.2.3. Funciones Medibles

Será necesario asimismo utilizar el concepto de *funciones medibles* [Bil95] para describir ciertos requisitos de las funciones de los modelos. Este concepto se describe por medio de las siguientes definiciones:

Definición 3.1. ***Preimagen.*** *Dada una función $f : A \to B$ y un conjunto $B_1 \subseteq B$, se define* preimagen $A_1 \triangleq f^{-1}(B_1)$ *del conjunto B_1 bajo f como:*

$$A_1 = \{a \in A \mid f(a) \in B_1\}$$

Definición 3.2. ***Función medible.*** *Dada una sigma–álgebra $\mathcal{A}$ sobre un conjunto A y un sigma–álgebra $\mathcal{B}$ sobre un conjunto B, una función $f : A \to B$ se dice que es* medible $\mathcal{A}/\mathcal{B}$ *cuando se cumple:*

$$A_1 = f^{-1}(B_1) \Rightarrow A_1 \subseteq A \ , \forall B_1 \subseteq B$$

La idea es que cuando una función medible genera un conjunto imagen medible, entonces asegura un conjunto preimagen medible. Estos conjuntos son medibles dado que pertenecen a sus correspondientes sigma–álgebra, y luego la función en sí misma es llamada *medible* en el contexto de $\mathcal{A}$ y $\mathcal{B}$. Por razones de brevedad, en general se omite la referencia a las sigma–álgebra, y f es simplemente llamada *función medible.*

3.3. Motivación

En esta sección se tratan los principales tópicos teóricos exponen la necesidad de reconsiderar al formalismo DEVS a la hora de modelar procesos estocásticos. Se discute una serie de preguntas abiertas que muestran la necesidad de un nuevo enfoque para solucionar los problemas.

3.3.1. DEVS con Números Aleatorios. DEVS–RND

Se plantea aquí una pregunta básica a modo de motivación para el análisis: ¿Se pueden utilizar generadores de números aleatorios como parte de las funciones DEVS?

Luego, con la intención de responder esta consigna, se estudiarán algunos modelos DEVS muy sencillos que incorporan el uso de números aleatorios, y se arribará a la conclusión de que no requiere demasiado esfuerzo encontrar rápidamente limitaciones importantes que invalidan la correctitud de los modelos.

En adelante, se hará referencia a DEVS con Números Aleatorios (DEVS–RND, por *DEVS with RND numbers*), llamando *modelo DEVS con Números Aleatorios (DEVS–RND)* a aquellos cuyas funciones dependen de (al menos una) variable aleatoria, y se hará foco principalmente en la función de transición interna para plantear casos de estudio.

Comenzando con un modelo básico DEVS–RND M_U que emite cada un segundo un nuevo número aleatorio entre 0 y 1 tomado de una distribución uniforme. Dicho modelo puede ser definido según:

$$M_U = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

donde

- $X = \oslash$ (el modelo no tiene entradas, es un generador)
- $Y = \Re_{[0,1]}$
- $S = \Re_{[0,1]} \times \Re_0^+$
- $\delta_{\text{int}}(a, \sigma) = (RND(0,1), 1)$ (siempre ejecuta una transición interna cada 1 segundo)
- $\delta_{\text{ext}}(s, e, x) = \oslash$ (el modelo no posee transiciones externas)
- $\lambda(s) = a$ (donde $a \in \Re_{[0,1]}$ es parte del estado utilizado para almacenar la última muestra aleatoria generada)
- $ta(s) = \sigma$ (donde $\sigma \in \Re_0^+$ es la parte del estado utilizada para almacenar el tiempo que debe transcurrir hasta la próxima transición interna)

Ahora se supondrá que en un momento dado el estado del modelo es $s = (0{,}5, 1)$. El próximo estado s' quedará determinado por $s' = \delta_{\text{int}}(0{,}5, 1)$ que es un valor indeterminado hasta que el experimento aleatorio descrito por RND(0,1) sea llevado a cabo. Esta situación muestra que δ_{int} *no es una función* (no asigna un valores únicos para un conjunto dado de variables de entrada). Luego, el modelo no es un modelo DEVS correcto (δ_{int} debe ser una función, de acuerdo a su definición formal).

El atajo que puede tomarse para solucionar esta simple situación, es reescribir $\delta_{\text{int}}(\cdot)$ [2] con una estructura apropiada que refleje matemáticamente una

[2] Se utiliza el símbolo $\cdot$ en el argumento de una función para denotar que dicha función depende de *alguna lista de argumentos*.

función, pero conservando la descripción estocástica [3] . Esto se logra gracias a incorporar la variable aleatoria como un argumento. Con este enfoque alternativo , puede escribirse:

$$\delta_{\text{int}}(a, \sigma, r) = (r, 1) \text{ with } r \sim U(0, 1)$$

En general, puede decirse que ahora la función de transición interna depende del estado s $\boldsymbol{y}$ de alguna variable aleatoria r con una distribución estadística dada, y que esta definición es matemáticamente correcta.

Al hacer esto, la función $\delta_{\text{int}}(s, r)$ calcula un estado $s \in S$ acorde a la elección de un número aleatorio r.

3.3.2. Generalidad de DEVS–RND

Ahora que se dispone de la nueva forma DEVS–RND con una función de transición interna correctamente definida, se plantea una nueva pregunta: ¿Es la estructura $\delta_{\text{int}}(s, r)$ la forma más genérica posible para elegir un elemento al azar del conjunto S ? La respuesta es *no*.

Dado que S en DEVS es un conjunto abstracto, puede consistir en una estructura muy compleja de tal manera que no sea posible asignar unívocamente números reales a cada elemento de S.

Por ejemplo, se puede pensar en un modelo DEVS–RND que elija cada un segundo una función $f_c : [0, 1] \to \Re$ tomada del espacio de todas las funciones continuas $C([0, 1])$, con $f_c \in C([0, 1])$. Luego, el modelo calcula el valor promedio de f_c sobre el intervalo $[0, 1]$ envía el resultado en forma de valor de salida.

En este modelo, el espacio de estados es infinito–dimensional, y dado que los números reales no pueden ser asignados unívocamente en este espacio, el modelo no puede ser definido utilizando la nueva estructura DEVS–RND. La única manera de elegir elementos aleatoriamente del espacio de funciones continuas es por medio del uso de espacios de probabilidad.

A pesar de que para la mayoría de las necesidades prácticas en simulación alcanza con elegir números reales (espacio infinito unidimensional), el objetivo en la presente Tesis es desarrollar un formalismo de modelado *genérico*.

3.3.3. Consistencia de DEVS–RND

Nuevamente se procede a formular una pregunta para motivar el análisis: ¿Es DEVS–RND consistente?

[3] En el trabajo de [Zei76] donde δ_{int} fue definida formalmente por primera vez se propone un ejemplo *Grocery Store* (almacén de alimentos) en donde una función $\Gamma : [0, 1] \to [0, 1]$ es utilizada como generador de números al azar para construir una función de transición interna del siguiente modo: $\delta_{\text{int}}(s) = f(\cdot, \Gamma(s))$. Esta última es una correcta definición DEVS, dado que $\Gamma(\cdot)$ representa un caso de *modelado determinístico* de un sistema estocástico. Estrictamente hablando, así definida δ_{int} no depende de una verdadera variable aleatoria, sino en cambio de una trayectoria determinista de valores Γ. En rigor, se acepta la suposición de que dicha trayectoria *imitará satisfactoriamente* las propiedades estadísticas del *verdadero* proceso aleatorio que se pretende modelar (es decir, la trayectoria es una secuencia pseudo–aleatoria). Con el propósito de lograr una representatividad precisa de modelado, se requiere que la función Γ provea la *"noción de un generador de números aleatorios* ideal*"*, y las sugerencias para formular este requisito en términos formales se ofrecen en [Zei76]. Sin embargo, δ_{int} sigue siendo determinista, ajustándose de este modo a la definición matemática de *función*.

La idea es saber si es consistente [4] o no incorporar variables aleatorias como argumentos en DEVS–RND tal como se propuso para la función δ_{int}.

Para contestar esta pregunta, será necesario presentar un segundo ejemplo sencillo. El siguiente modelo M_V selecciona un número aleatorio $r \sim U(0,1)$ cada un segundo, del mismo modo que lo hacía el modelo M_U, pero esta vez M_V también calcula el promedio de la cantidad de veces que el resultado r del experimento aleatorio pertenece a un subconjunto dado que se denominará $V \subset \Re_{[0,1]}$. El nuevo modelo puede definirse formalmente como sigue:

$$M_V = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

donde

- $X = \oslash$ (el modelo no posee entradas)
- $Y = \Re_{[0,1]}$
- $S = \Re_{[0,1]} \times \mathbb{N} \times \Re_0^+$
- $\delta_{\text{int}}(v, n, \sigma, r) = (v', n+1, 1)$ with $v' = \begin{cases} \frac{v \cdot n + 1}{n+1} & \text{, si } r \in V \subset \Re_{[0,1]} \\ \frac{v \cdot n}{n+1} & \text{, en cualquier otro caso.} \end{cases}$
  (donde n es la parte del estado usada para contar el número de experimentos ejecutados)
- $\delta_{\text{ext}}(s, e, x) = \oslash$ (el modelo no tiene transiciones externas)
- $\lambda = v$ (donde $v \in \Re_{[0,1]}$ es la parte del estado usada para almacenar el número promedio de éxitos para $V \subset \Re_{[0,1]}$)
- $ta = \sigma$ (donde $\sigma \in \Re_0^+$ es la parte del estado usada para almacenar el tiempo remanente hasta la próxima transición interna)

Es evidente que cuando n tiende a infinito, el valor de salida converge a la probabilidad de que un número real entre 0 y 1 pertenezca a V.

Sin embargo, si V fuese elegido como el conjunto de Vitali [Vit05], dicha probabilidad no existe, ya que los conjuntos de tipo Vitali son no medibles. Un conjunto de Vitali es un ejemplo de un conjunto de números reales que no es medible–Lebesgue, siendo entonces *inconsistente* hablar sobre su longitud.

Consecuentemente, a pesar de que el modelo M_V es aparentemente correcto de acuerdo a la nueva *solución* de agregar r a los argumentos de δ_{int}, esto es matemáticamente inconsistente. Es imposible generar un número real aleatorio y luego preguntar si pertenece o no a un conjunto de Vitali, dado que este último no tiene medida sobre los número reales.

Finalmente, la respuesta a la pregunta original es nuevamente *no*. Dicho de otro modo, es posible encontrar modelos inconsistentes cuando se agregan variables aleatorias directamente en los argumentos de las funciones de transición.

[4] Se hace referencia al concepto de *consistencia* informalmente, para denotar la situación en la cual un modelo no conduce a un contradicción matemática.

3.3.4. Clausura bajo acoplamiento de DEVS–RND

Ahora se procederá a verificar DEVS–RND con respecto a las propiedades teóricas fundamentales de DEVS clásico, comenzando con la clausura bajo acoplamiento. Para ello, se formula la siguiente pregunta: ¿El acoplamiento de dos o más modelos DEVS–RND consistentes , produce siempre un modelo equivalente DEVS–RND consistente?

En otras palabras, se plantea la pregunta de si el agregado de variables aleatorias a un modelo DEVS puede o no afectar la garantía de la propiedad de clausura bajo acoplamiento.

Considérese nuevamente el modelo M_U presentado en la Sección 3.3.1, expresado acorde a la forma DEVS–RND:

$$M_U = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

donde

- $X = \oslash$ (el modelo no tiene entradas)
- $Y = \Re_{[0,1]}$
- $S = \Re_{[0,1]} \times \Re^{+}$
- $\delta_{\text{int}}(a, \sigma, r) = (r, 1)$ with $r \sim U(0, 1)$ (una transición interna por segundo)
- $\delta_{\text{ext}}(s, e, x) = \oslash$ (el modelo no tiene transiciones externas)
- $\lambda(s) = a$
- $ta(s) = \sigma$

Luego, considérese que se conecta la salida de M_U a la entrada de un modelo DEVS determinístico M_A, el cual calcula (en su estado) el promedio de los valores recibidos por su entrada cuando estos valores pertenecen al conjunto de Vitali $V \subset \Re_{[0,1]}$:

$$M_A = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

donde

- $X = \Re_{[0,1]}$
- $Y = \Re_{[0,1]}$
- $S = \Re_{[0,1]} \times \mathbb{N} \times 0, \infty$
- $\delta_{\text{int}}(v, n, \sigma) = (v, n, \infty)$
- $\delta_{\text{ext}}(v, n, \sigma) = (v', n + 1, 0)$ with $v' = \begin{cases} \frac{v \cdot n + 1}{n+1} & \text{, si } r \in V \subset \Re_{[0,1]} \\ \frac{v \cdot n}{n+1} & \text{, en cualquier otro caso.} \end{cases}$
- $\lambda(v, n, \sigma) = v$
- $ta(v, n, \sigma) = \sigma$

Ambos modelos DEVS son perfectamente consistentes. El primero es uno muy simple, y el segundo es sencillamente un clásico modelo DEVS determinístico. Sin embargo, el acoplamiento de ambos modelos consistentes M_U y M_A resulta en un modelo que se comporta exactamente como el modelo atómico M_V estudiado en la Sección 3.3.3, que es claramente inconsistente.

Nuevamente, se obtuvo un *no* como respuesta a la pregunta planteada: procediendo de este modo no es posible garantizar que se conserve la propiedad de clausura bajo acoplamiento y al mismo tiempo preservar la consistencia.

3.3.5. Legitimidad de DEVS–RND

Otra propiedad teórica importante de DEVS clásico es la que trata con la legitimidad. Entonces, la nueva pregunta que se plantea es la siguiente: ¿Es posible aplicar el concepto clásico de legitimidad de DEVS a modelos DEVS–RND?

Considérese ahora el siguiente modelo DEVS–RND, el cual simplemente arroja el valor 1 en su salida cada una cantidad de tiempo elegida aleatoriamente:

$$M_L = (X, Y, S, \delta_{\text{int}}, \delta_{\text{ext}}, \lambda, ta)$$

donde

- $X = \oslash$ (el modelo no tiene entradas)
- $Y = 1$
- $S = \Re_{[0,1]}$
- $\delta_{\text{int}}(\sigma, r) = r$ con $r \sim U(0, 1)$
- $\delta_{\text{ext}} = \oslash$ (el modelo no tiene transiciones externas)
- $\lambda(s) = 1$ (el modelo siempre emite el valor 1
- $ta(s) = \sigma$

¿Es legítimo este modelo? Es decir, ¿Experimentará siempre una cantidad finita de transiciones de estados en cualquier intervalo finito de tiempo?

Notar que la siguiente secuencia de valores de r es perfectamente posible y resulta en una trayectoria ilegítima de estados del modelo: $\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, ...\}$. De igual modo, la secuencia $\{0, 0, 0, 0, ...\}$ puede ocurrir, conduciendo a M_L a una condición ilegítima en el sentido de DEVS clásico.

Aun así, es sencillo probar que la *probabilidad* de obtener una secuencia ilegítima es 0, obteniendo así un modelo legítimo *en cierto sentido*. Sin embargo, para hacer esta afirmación, se deberá redefinir el concepto de legitimidad adaptado al contexto de modelos estocásticos.

3.3.6. La necesidad de un nuevo formalismo

Por último, tras considerar las últimas preguntas formuladas, se puede arribar a una última pregunta central: ¿Es necesario un nuevo formalismo para definir modelos DEVS estocásticos?

Los ejemplos sencillos examinados antes en este capítulo hicieron evidente que incorporar propiedades estocásticas a modelos DEVS puede redundar en inconsistencias inadmisibles.

Otros formalismos de eventos discretos (como Redes de Petri, Autómatas de Estados Finitos, statecharts, etc.) trabajan con conjuntos de estados finitos (contables). Luego, incorporar en ellos características estocásticas es un procedimiento inmediato y no sufre de contravenciones teóricas.

DEVS, en cambio, admite conjuntos de estados arbitrarios. Desafortunadamente, la única manera de trabajar consistentemente con procesos aleatorios operando sobre conjuntos arbitrarios es por medio del uso de espacios de probabilidad. Todos los problemas que surgieron al postular las preguntas que guiaron las secciones precedentes tienen una raíz en común, que consiste en haber intentado incorporar propiedades estocásticas sin tener en cuenta la generalidad del espacio de estados.

Luego, la respuesta a la última pregunta que permanece abierta es esta vez *sí*. Es evidente que un formalismo estocástico y genérico DEVS debe ser desarrollado si se pretende incorporar elementos de la teoría de Espacios de Probabilidad.

En lo sucesivo presentaremos este nuevo formalismo, denominado STDEVS.

En la práctica, las computadoras utilizan un número finito de bits para la representación de datos. Así, cuando se simula un modelo DEVS, el conjunto de estados S se representa como un conjunto finito (evitando así los problemas discutidos en este capítulo). Sin embargo, cabe notar que una de las características centrales de DEVS es que el formalismo distingue *explícitamente* la actividad de *Modelado* de aquella de *Simulación*. En esta línea, no sería apropiado hacer ninguna suposición previa acerca de la finitud del espacio de estados, que consiste solamente en una consecuencia particular de la implementación digital de la actividad de *simulación*).

Para redondear, el beneficio de un nuevo formalismo STDEVS será proveer la teoría que falta para obtener una descripción genérica, consistente, cerrada, legítima y correcta para modelos DEVS estocásticos.

3.4. El formalismo STDEVS

En esta sección se presenta el nuevo formalismo STDEVS. Luego de una descripción informal de la idea central detrás de STDEVS, se presenta aquí la definición formal de un modelo atómico STDEVS y de un modelo acoplado STDEVS.

3.4.1. Conceptos

La estrategia utilizada consiste básicamente en tomar la definición determinista de DEVS (manteniendo la esencia de su estructura de modelo), para luego derivar de ella una nueva estructura de modelo estocástico que reemplace la forma en la que se describen las dinámicas de transición de estados.

Concretamente, las funciones deterministas δ_{int} y δ_{ext} deberán ser reemplazadas por sus contrapartes probabilísticas en términos de la teoría de Espacios de Probabilidad. La nueva estructura de modelo estocástico será referida como STDEVS, y deberá garantizar tanto la correctitud del uso de funciones aleatorias

dentro de la descripción del comportamiento de los modelos como la correctitud de acoplar modelos estocásticos y deterministas.

Como decisión de diseño, se decide utilizar las funciones de *transición de estados* de un modelo atómico como aquellos elementos matemáticos candidatos para incorporarles comportamiento estocástico. Luego, se pensará acerca de cada *transición de estado* como un *experimento aleatorio* cuyo resultado determinará el próximo estado al cual evoluciona el modelo luego de realizado dicho experimento.

Dado que en las transiciones internas de DEVS el estado futuro $\tilde{s}$ queda determinado por el estado presente s, se deberá definir un espacio de probabilidad para cada estado presente s (es decir, se deberá asignar probabilidad a cada posible transición partiendo de s y resultando en $\tilde{s}$).

De modo similar, en las transiciones externas de DEVS, el estado futuro $\tilde{s}$ queda determinado por el estado presente s, el tiempo transcurrido e y el evento de entrada x. Luego, en STDEVS se deberá definir un espacio de probabilidad para cada triplete (s, e, x) (es decir, se deberá asignar probabilidad a cada posible transición partiendo de (s, e, x) y resultando en $\tilde{s}$).

Ambas transiciones DEVS y experimentos aleatorios STDEVS producen un nuevo $\tilde{s} \in S$ como resultado de una transición de estados. Luego, el vínculo natural entre ambos tipos de estructuras (DEVS y STDEVS) es el conjunto S. El mismo, deberá actuar como el *conjunto de estados* para ambas estructuras, y como el *espacio muestral* S_{sp} requerido para construir una descripción estocástica en términos de espacios de probabilidad en el contexto de la estructura STDEVS.

Dado que el conjunto de estados S puede ser continuo, proveer una medida de probabilidad de tipo estado–a–estado sería inútil en la mayoría de los casos, ya que la misma sería 0 para la mayoría de los estados posibles $\tilde{s}$. Cuando S es continuo, se requiere un espacio muestral continuo y una distribución de probabilidad continua para medir la probabilidad los posibles estados futuros tomados de S.

Esta descripción viene dada por funciones de densidad de probabilidades *pdf* (probabilidad density function), las cuales deben ser integradas sobre un intervalo para producir una *medida de probabilidad* para el *evento* de que el resultado del *experimento aleatorio* quede comprendido en dicho intervalo.

En este escenario, si se elige un estado futuro *individual* $\tilde{s}$ y se calcula la integral de la *pdf* sobre un punto, el resultado será siempre nulo. Luego, un enfoque genérico deberá tomar en cuenta una medida de probabilidad describiendo las chances de que estando el sistema en un estado s *evolucione hacia* ***conjuntos*** $G \subset S$ luego de que se ejecute una transición interna (y razonando análogamente para el caso de una transición externa).

Para no perder generalidad, cada transición en STDEVS debe ser vista como un experimento aleatorio independiente. Luego, para una transición interna, dependiendo del estado actual,se debería poder asignar probabilidades para futuros estados perteneciendo a distintos subconjuntos de S. En otras palabras, la colección $\mathcal{G}$ (que contiene los subconjuntos de S a los cuales se le asignan medidas de probabilidad cuando el estado es s) debe depender de s.

Luego, deberemos definir una función recolectora de conjuntos $\mathcal{G}_{int}(s)$ la cual, junto con un espacio muestral S dado y una función de probabilidad, definan un espacio de probabilidad que reemplace la función de transición interna determinista de DEVS. Una afirmación similar puede hacerse para la transición

externa, en donde la función recolectora de conjuntos $\mathcal{G}_{ext}(s, e, x)$ dependerá también del tiempo transcurrido y del evento de entrada.

El enfoque que se acaba de proponer se basa en utilizar descripciones estocásticas *únicamente en las funciones de transición de estados*. Esto permite reducir notablemente la complejidad de la descripción de la estructura estocástica, sin perder generalidad. La aleatoriedad incorporada en las funciones λ y ta pueden ser incorporadas teniendo en cuenta que dichas funciones dependen del estado $s \in S$. Ya que s será elegido siempre por experimentos aleatorios en las transiciones interna y externa, la aleatoriedad de las demás funciones pueden ser modeladas como parte de la información almacenada en el estado.

Una forma directa de lograr esto es definir al estado como $s = (\hat{s}, \sigma, \ell)$ y luego hacer $ta(s) = \sigma$ y $\lambda(s) = \ell$. Luego, puede definirse un experimento aleatorio para elegir σ y ℓ en las transiciones interna y externa, obteniéndose un comportamiento aleatorio arbitrario para $\lambda(s)$ y $ta(s)$ (a pesar de que sus definiciones matemáticas permanezcan deterministas).

3.4.2. Definición de STDEVS

Un modelo STDEVS tiene la siguiente estructura:

$$M_{ST} = (X, Y, S, \mathcal{G}_{int}, \mathcal{G}_{ext}, P_{int}, P_{ext}, \lambda, ta) \tag{3.1}$$

donde

- X, Y, S preservan la definición original que poseen en DEVS.
- $\mathcal{G}_{int}, \mathcal{G}_{ext}, P_{int}, P_{ext}$ son funciones nuevas que reemplazan la funcionalidad de las originales δ_{ext} y δ_{int} en DEVS.
- λ, ta extienden la definición original que poseen en DEVS con el requerimiento adicional de que sean *funciones medibles*.

$\mathcal{G}_{int} : S \to 2^S$ es una función que asigna una colección de conjuntos medibles $\mathcal{G}_{int}(s) \subseteq 2^S$ a cada estado s. Dado un estado s, la colección $\mathcal{G}_{int}(s)$ contiene a todos los subconjuntos medibles de S a los cuales podría pertenecer el estado futuro $\tilde{s}$ con una probabilidad conocida, dada por la función $P_{int} : S \times 2^S \to [0, 1]$. Cuando el sistema está en el restado s la probabilidad de que una transición interna lo lleve a un conjunto $G \in \mathcal{G}_{int}(s)$ es calculada por $P_{int}(s, G)$.

Llamando $\mathcal{F}_{int}(s) \triangleq \mathcal{M}(\mathcal{G}_{int}(s))$ a la mínima sigma–álgebra generada por $\mathcal{G}_{int}(s)$, el triplete $(S, \mathcal{F}_{int}(s), P_{int}(s, \cdot))$ es un espacio de probabilidad para cada estado $s \in S$.

Procediendo del mismo modo, $\mathcal{G}_{ext} : S \times \Re_0^+ \times X \to 2^S$, es una función que asigna una colección de conjuntos $\mathcal{G}_{ext}(s, e, x) \subseteq 2^S$ a a cada triplete (s, e, x). Dado un estado s y un tiempo transcurrido e, si llega un evento externo con valor x, $\mathcal{G}_{ext}(s, e, x)$ contendrá todos los conjuntos medibles de S a los cuales podría pertenecer el estado futuro $\tilde{s}$, con una probabilidad conocida dada por $P_{ext} : S \times \Re_0^+ \times X \times 2^S \to [0, 1]$.

Llamando $\mathcal{F}_{ext}(s, e, x) \triangleq \mathcal{M}(\mathcal{G}_{ext}(s, e, x))$ a la mínima sigma–álgebra generada por $\mathcal{G}_{ext}(s, e, x)$, el triplete $(S, \mathcal{F}_{ext}(s, e, x), P_{ext}(s, e, x, \cdot))$ es un espacio de probabilidad para cada triplete (s, e, x).

3.4.3. Acoplamiento en STDEVS

Procediendo de modo similar lo propuesto en la Sección 2.2 para acoplamiento DEVS, se define que los modelos STDEVS pueden ser acoplados modularmente de tal modo que un modelo acoplado STDEVS N queda definido por la siguiente estructura:

$$N = (X_N, Y_N, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\}, Select)$$

donde los componentes son idénticos a sus homólogos en la definición de DEVS, excepto que los componentes M_d son ahora estructuras STDEVS.

El nuevo requisito incorporado para las funciones λ y ta acerca de su medibilidad (para un modelo atómico STDEVS), garantizará la deseada medibilidad de $\mathcal{G}_{int}$ y $\mathcal{G}_{ext}$ para modelos acoplados STDEVS. Se regresará sobre este punto con mayor detalle en la Sección 3.5.1.

3.5. Propiedades de STDEVS

En la presente sección se estudiarán las principales propiedades de STDEVS. Primero se mostrará que STDEVS es cerrado bajo acoplamiento. (es decir, el acoplamiento de modelos atómicos STDEVS es equivalente a un nuevo modelo atómico DEVS equivalente). Luego, se redefinirá el concepto de legitimidad DEVS para el nuevo contexto de STDEVS.

3.5.1. Clausura Bajo Acoplamiento

Se demostrará que un modelo STDEVS $N = \langle X_N, Y_N, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\}, Select\rangle$ siendo $M_d \in \{M_d\}$ modelos atómicos STDEVS para todo d, define un modelo atómico equivalente STDEVS , verificando así la clausura bajo acoplamiento.

Para lograr esto, se encontrará un modelo atómico STDEVS $M_{ST} = (X, Y, S_N, \mathcal{G}_{int_N}, \mathcal{G}_{ext_N}, P_{int_N}, P_{ext_N}, \lambda, ta)$ definido por la expresión de acoplamiento N.

Se comienza definiendo las relaciones que son compartidas con la prueba clásica para DEVS determinista [ZKP00]:

- $X = X_N, Y = Y_N$
- $S_N = \underset{d \in D}{\times} \{(s_d, e_d)\}$ con $s_d \in S_d, e_d \in \Re$. Cada componente de S_N tiene la forma $s_N = (\ldots, (s_d, e_d), \ldots)$.
- $ta(s_N) = min\{\sigma_d \mid d \in D\}$, con $\sigma_d = t_{a_d}(s_d) - e_d$.
- $d^* = Select(IMM(s_N))$, donde IMM es el conjunto de submodelos con mínimo tiempo remanente hasta su próximo evento, es decir, $IMM = \{d | \sigma_d = ta(s_N)\}$.
- $\lambda(s_N) = \begin{cases} Z_{d^*,N}(\lambda_{d^*}(s_{d^*})) & \text{if } d^* \in I_N, \\ \oslash & \text{en cualquier otro caso.} \end{cases}$

Luego, se requerirá obtener los espacios de probabilidad que representarán los fenómenos estocásticos del modelo acoplado, como resultado del comportamiento estocástico de sus componentes.

En primer lugar, para transiciones internas, se definirá la función recolectora de conjuntos:

$$\mathcal{G}_{int_N}(s_N) \triangleq \underset{d \in D}{\times}(\mathcal{G}_d \times \{\tilde{e}_d\})$$

donde

$$\mathcal{G}_d = \begin{cases} \mathcal{G}_{int}(s_{d^*}) & \text{if } d = d^*, \\ \mathcal{G}_{ext}(s_d, \hat{e}_d, x_d) & \text{if } x_d \neq \oslash, \\ \{s_d\} & \text{en cualquier otro caso.} \end{cases}$$

con

$$x_d = \begin{cases} Z_{d^*,d}(\lambda_{d^*}(s_{d^*})) & \text{if } d^* \in I_d, \\ \oslash & \text{en cualquier otro caso.} \end{cases}$$

$$\tilde{e}_d = \begin{cases} 0 & \text{if } d = d^* \text{ or } x_d \neq \oslash, \\ \hat{e}_d & \text{en cualquier otro caso.} \end{cases}$$

and

$$\hat{e}_d = e_d + ta_{d^*}(s_{d^*}) - e_{d^*}$$

Luego, los conjuntos $G_N \in \mathcal{G}_{int_N}(s_N)$ tendrán la forma $G_N = (\ldots(G_d, \{e_d\}), \ldots)$ verificando $G_N \subseteq S_N$.

También se llamará $\mathcal{F}_{int_N}(s_N) \triangleq \mathcal{M}(\mathcal{G}_{int_N}(s_N))$ a la mínima sigma–álgebra generada por $\mathcal{G}_{int_N}(s_N)$. Luego, la medida de probabilidad $P_{int_N} : S_N \times 2^{S_N} \to [0, 1]$ para el proceso de transición interna en N, se define como:

$$P_{int_N}(s_N, G_N) \triangleq P_{int_{d^*}}(s_{d^*}, G_{d^*}) \prod_{d|x_d \neq \oslash} P_{ext_d}(s_d, \hat{e}_d, x_d, G_d)$$

pudiéndose verificar que el triplete $(S_N, \mathcal{F}_{int_N}(s_N), P_{int_N}(s_N, \cdot))$ es un espacio de probabilidad.

De modo similar, para transiciones externas, se definirá la función recolectora de conjuntos:

$$\mathcal{G}_{ext_N}(s_N, e, x_N) \triangleq \underset{d \in D}{\times}(\mathcal{G}_d \times \{\tilde{e}_d\})$$

donde

$$\mathcal{G}_d = \begin{cases} \mathcal{G}_{ext}(s_d, \hat{e}_d, x_d) & \text{if } x_d \neq \oslash, \\ \{s_d\} & \text{en cualquier otro caso.} \end{cases}$$

$$\tilde{e}_d = \begin{cases} 0 & \text{if } x_d \neq \oslash, \\ \hat{e}_d & \text{en cualquier otro caso.} \end{cases}$$

con

$$x_d = \begin{cases} Z_{N,d}(x_N) & \text{if } N \in I_d, \\ \oslash & \text{en cualquier otro caso.} \end{cases}$$

y

$$\hat{e}_d = e_d + e$$

Los conjuntos $G_N \in \mathcal{G}_{ext_N}(s_N, e, x_N)$ también tendrán la forma $G_N = (\ldots(G_d, \{e_d\}), \ldots)$ y verificarán $G_N \subseteq S_N$.

Nuevamente, se definirá $\mathcal{F}_{ext_N}(s_N, e, x_N) \triangleq \mathcal{M}(\mathcal{G}_{ext_N}(s_N, e, x_N))$ a la mínima sigma–álgebra generada por $\mathcal{G}_{ext_N}(s_N, e, x_N)$. Luego, la medida de probabilidad para el proceso de transición externa en N, $P_{ext_N} : S_N \times \Re \times X \times 2^{S_N} \to [0, 1]$ se define como:

$$P_{ext_N}(s_N, e, x_N, G_N) = \prod_{d|x_d \neq \emptyset} P_{ext_d}(s_d, \hat{e}_d, x_d, G_d)$$

siendo el triplete $(S_N, \mathcal{F}_{ext_N}(s_N, e, x_N), P_{ext_N}(s_N, e, x_N, \cdot))$ un espacio de probabilidad.

Notar que está garantizado que las funciones $\lambda(s_N)$ y $ta(s_N)$ resultantes del procedimiento de acoplamiento son funciones medibles. Esto es así porque cualquier número finito de operaciones involucrando funciones medibles sobre conjuntos medibles resulta siempre en funciones y/o conjuntos medibles. El mismo argumento puede ser aplicado para mostrar que los subconjuntos $G_N \in \mathcal{G}_{int_N}(\cdot)$ y $G_N \in \mathcal{G}_{ext_N}(\cdot)$ serán conjuntos medibles.

Esta prueba constructiva demuestra que el modelo genérico acoplado STDEVS N es equivalente al modelo atómico STDEVS M_{ST}. Luego, es posible acoplar modelos STDEVS de modo jerárquico, encapsulando modelos acoplados complejos y acoplándolos con otros modelos tanto acoplados como atómicos.

Esta propiedad es análoga a la del formalismo DEVS clásico (determinista).

3.5.2. Legitimidad

La legitimidad en DEVS determinista es una propiedad que asegura que el modelo no puede ejecutar un número infinito de transiciones de estado en un intervalo finito de tiempo. En STDEVS, esta propiedad debe ser redefinida para expresar, en cambio, que la *probabilidad* de experimentar un número infinito de transiciones en un intervalo finito sea *cero*.

Dado un modelo STDEVS, se considerará una función $P_k : S \times \Re^+ \to [0, 1]$, tal que $P_k(s, z)$ evalúe la probabilidad de que partiendo del estado $s \in S$, luego de k transiciones internas, el sistema acumule un tiempo transcurrido menor o igual que z.

Luego se define la *legitimidad* de STDEVS como sigue:

Definición 3.3. *Un modelo STDEVS se dice legítimo cuando verifica:*

$$\lim_{k \to \infty} P_k(s, z) = 0, \forall z < \infty, \forall s \in S \tag{3.2}$$

En otras palabras, un modelo STDEVS es legítimo cuando, partiendo desde cualquier estado s, la probabilidad de acumular un tiempo transcurrido finito luego de una secuencia infinita de cambios de estado es 0.

Por ejemplo, de acuerdo a esta definición, el modelo M_L presentado en la Sección 3.3.5 es legítimo, ya que verifica la Ec.(3.2).

Notar que en este modelo se cumple:

$$P_k(s, z) = P\left(\sum_{i=1}^{k} r_i < z\right) \ \forall s \in S$$

donde los r_i son k números aleatorios uniformemente distribuidos entre 0 y 1. Dado cualquier número $\varepsilon > 0$ tan pequeño como se desee, la Ley (débil) de los Grandes Números establece que

$$\lim_{k\to\infty} P\left(|\sum_{i=1}^{k} \frac{r_i}{k} - \frac{1}{2}| < \varepsilon\right) = 1$$

de lo cual puede ser fácilmente deducido que

$$\lim_{k\to\infty} P\left(\sum_{i=1}^{k} r_i > k(\frac{1}{2} - \varepsilon)\right) = 1$$

y luego

$$\lim_{k\to\infty} P\left(\sum_{i=1}^{k} r_i < k(\frac{1}{2} - \varepsilon)\right) = 0$$

Ahora, tomando $\varepsilon < 1/2$, el término $k(1/2 - \varepsilon)$ resulta mayor que cualquier z a medida que k tiende a ∞. Entonces,

$$\lim_{k\to\infty} P\left(\sum_{i=1}^{k} r_i < z\right) = 0$$

y así el modelo M_L verificará la condición de legitimidad de STDEVS.

Para completar la Definición 3.3, se deducirá una expresión general para $P_k(s, z)$ en términos de las funciones características de STDEVS P_{int} y ta. Notar que la construcción que sigue para $P_k(s, z)$ es únicamente un ejemplo entre muchos otros posibles procedimientos que podrían ser propuestos para verificar la Definición 3.3.

Se comenzará seleccionando un escalar positivo pequeño ε, con $0 < \varepsilon \ll 1$ y definiendo los conjuntos $S_{x,\varepsilon}$ compuestos por aquellos estados que verifiquen un criterio de acotamiento para sus valores de avance de tiempo, de acuerdo a:

$$S_{x,\varepsilon} \triangleq \{s \mid x \leq ta(s) < x + \varepsilon\} \tag{3.3}$$

Ahora, se incorporará la función $p_k(s, z, \varepsilon)$ que evalúa la probabilidad de que el sistema, partiendo del estado s, acumule un tiempo transcurrido dado en el intervalo $[z, z + \varepsilon)$ luego de k transiciones internas. Para $k = 1$, y teniendo en cuanta la función P_{int} y los conjuntos $S_{x,\varepsilon}$ definidos antes, sigue entonces que:

$$p_1(s, z, \varepsilon) \triangleq P_{int}(s, S_{z,\varepsilon}) \tag{3.4}$$

Supóngase ahora que la expresión de p_k es conocida para ciertos valores de k. Para encontrar la expresión de p_{k+1}, primero se definirá $R_k(z, x, c, \varepsilon)$ como el conjunto de los estados $s \in S_{x,\varepsilon}$ con probabilidad $[c, c + \varepsilon)$ de acumular un tiempo transcurrido entre z y $z + \varepsilon$ luego de k transiciones. Formalmente,

$$R_k(z, x, c, \varepsilon) \triangleq \{s \in S_{x,\varepsilon} \mid c \leq p_k(s, x, z) < c + \varepsilon\} \tag{3.5}$$

Luego, asumiendo que ε es lo suficientemente pequeño, es posible evaluar $p_{k+1}(s, z, \varepsilon)$ como la probabilidad de realizar un paso de duración comprendida entre x y $x+\varepsilon$ seguido de k pasos con una duración total comprendida $z-x$ y $z-$

$x+\varepsilon$. Esta probabilidad puede ser calculada como la suma de las probabilidades de pasar por los distintos conjuntos disjuntos de estados $R_k(z-x,x,c,\varepsilon)$ con diferentes valores de c (entre 0 y 1) y x (entre 0 y z). Esto es,

$$p_{k+1}(s,z,\varepsilon) = \sum_{m_x=0}^{[z/\varepsilon]} \sum_{m_c=0}^{[1/\varepsilon]} P_{int}(s, R_k(z-\tilde{x},\tilde{x},\tilde{c},\varepsilon)).\tilde{c} \tag{3.6}$$

con

$$\tilde{x} = m_x\varepsilon, \tilde{c} = m_c\varepsilon \tag{3.7}$$

Se arribó entonces a una expresión genérica para p_k con $k \geq 1$. Luego, es posible construir $P_k(s,z)$ de acuerdo a:

$$P_k(s,z) = \lim_{\varepsilon\to 0} \sum_{m_x=0}^{z/\varepsilon} p_k(s,\tilde{x},\varepsilon) \tag{3.8}$$

en caso de que dicho límite exista [5].

Esta expresión, junto con Ec.(3.2), expresa una condición de legitimidad para STDEVS como una función dependiente de $P_{int}(s,\cdot)$ y $ta(s)$.

3.6. DEVS, funciones RND y STDEVS

En la presente sección se estudiarán los vínculos entre DEVS determinista y STDEVS (estocástico). En primer lugar se mostrará que es posible construir modelos STDEVS utilizando modelos DEVS deterministas equipados con funciones RND en sus funciones de transición. Luego, se mostrará que los modelos DEVS con funciones medibles son casos particulares de STDEVS y que esta propiedad habilita formalmente a acoplar modelos DEVS y STDEVS.

3.6.1. Modelos DEVS con funciones RND

Se mostrará que un modelo de tipo DEVS cuyas funciones de transición dependan de variables aleatorias (típicamente generadas utilizando funciones RND) definen, bajo ciertas condiciones, un modelo STDEVS.

Luego, en primer lugar, quedará claro que STDEVS puede representar cualquier modelo DEVS estocástico *práctico* y *consistente* definido por el método usual de utilizar funciones RND. En segundo lugar, se verá que esta propiedad permite definir y simular modelos STDEVS de una manera simple y directa, prescindiendo de utilizar espacios de probabilidad que agregan complejidad a la definición de la estructura del modelo, a su notación formal, y a su terminología.

Tal como se hizo antes se llamará modelo DEVS–RND a aquellos modelos DEVS cuyas funciones de transición dependan de funciones RND [6].

[5] en cualquier otro caso, cualquier otro procedimiento válido para encontrar una expresión para $P_k(s,z)$ acorde a la Definición 3.3 puede ser utilizado.

[6] Cabe distinguir el concepto de DEVS–RND (esto es una definición con fines de modelado matemático) del concepto usado en el trabajo original de Ziegler que utiliza un generador de números pseudo–aleatorios en una función de transición [Zei76] (esto es una especificación con fines prácticos de simulación en una computadora). Eventualmente, pero no necesariamente, un modelo DEVS–RND dado podrá ser simulado utilizando secuencias pseudo–aleatorias en funciones de transición DEVS y ofrecer resultados estadísticamente satisfactorios.

Asimismo, se dirá que un modelo DEVS–RND es medible cuando todas sus funciones (δ_{int}, δ_{ext}, ta, λ) son medibles sobre sus conjuntos correspondientes.

Teorema 3.1. *Un modelo DEVS–RND medible* $M_D = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta)$ *en el cual sus funciones de cambio de estado* δ_{int} *y* δ_{ext} *dependen dinámicamente de un experimento aleatorio por medio de una variable aleatoria* r *(es decir,* $\delta_{int} = \delta_{int}(s, r)$ *y* $\delta_{ext} = \delta_{ext}(s, e, x, r)$*) con* $r \in R \subseteq \Re^n$ *caracterizada por una medida de probabilidad* $P(r \in B \mid B \in \mathcal{B} \subseteq 2^R)$*, define un modelo equivalente STDEVS* [7]*.*

Demostración. Se deberá obtener un modelo STDEVS $M_{ST} = (X, Y, S, \mathcal{G}_{int}, \mathcal{G}_{ext}, P_{int}, P_{ext}, \lambda, ta)$ equivalente a M_D, asumiendo que X, Y, S, λ, ta son idénticas para M_D y M_{ST}. Luego, será necesario únicamente encontrar $\mathcal{G}_{int}, \mathcal{G}_{ext}, P_{int}$ y P_{ext}.

Se comienza definiendo la función recolectora de conjuntos $\mathcal{G}_{int}(s)$ en relación a la sigma–álgebra $\mathcal{B}$ del experimento aleatorio. Para cada conjunto $B \in \mathcal{B}$ y para cada estado $s \in S$, se define el *conjunto preimagen* $G_{s,B} \subseteq S$ de acuerdo a:

$$\hat{s} \in G_{s,B} \iff \exists\, r \in B / \delta_{\text{int}}(s, r) = \hat{s}$$

Dado que δ_{int} es una función medible, el conjunto B es medible, luego el conjunto $G_{s,B}$ resulta también medible. Entonces, se define $\mathcal{G}_{int}(s)$ como:

$$\mathcal{G}_{int}(s) \triangleq \{G_{s,B} | B \in \mathcal{B}\}$$

Luego, para un sistema estando en el estado s, la probabilidad de se produzca una transición a un nuevo estado perteneciendo a $G_{s,B} \in \mathcal{G}_{int}(s)$ será:

$$P_{int}(s, G_{s,B}) = P(r \in B)$$

Luego, para cada estado $s \in S$, la función $P_{int}(s, \cdot)$ será una medida de probabilidad en el espacio medible $(S, \mathcal{F}_{int}(s))$, siendo $\mathcal{F}_{int}(s) = \mathcal{M}(\mathcal{G}(s))$ la mínima sigma–álgebra generada por $\mathcal{G}_{int}(s)$. Esto se comprueba por medio de la verificación de los siguientes axiomas.

1. $P_{int}(s, G_{s,B}) \geq 0$ dado que $P_{int}(s, G_{s,B}) = P(r \in B) \geq 0$.
2. $P_{int}(s, S) = 1$, dado que $\delta_{\text{int}}(s, r) \in S, \forall s, r$.
3. Sea $B_1, B_2 \in \mathcal{B}$. Luego, si $G_{s,B_1} \cap G_{s,B_2} = \oslash \Rightarrow B_1 \cap B_2 = \oslash$. Entonces, lo siguiente es verdadero: $P_{int}(s, G_{s,B_1} \cup G_{s,B_2}) = P(r \in B_1 \cup B_2) = P(r \in B_1) + P(r \in B_2) = P_{int}(s, G_{s,B_1}) + P_{int}(s, G_{s,B_1})$

Hasta aquí se obtuvo $\mathcal{G}_{int}$ y P_{int} para el modelo STDEVS M_{ST} partiendo de la definición del modelo DEVS–RND M_D y la condición de aleatoriedad incorporada en $\delta_{int}(s, r)$.

para el caso de $\mathcal{G}_{ext}$ y P_{ext} se procede análogamente, esta vez reemplazando el estado s por el triplete (s, e, x) para el análisis. Con esto concluye la prueba. □

[7] Se llamará $\mathcal{B}$ a la sigma–álgebra donde la función P está definida.

Considérese ahora el caso particular de tener $r \in R = [0,1]^n \subset \Re^n$ con una distribución uniforme. Se dirá que r está uniformemente distribuida cuando cada componente de r tenga una distribución uniforme sobre el intervalo $[0,1]$:

$$r_i \sim U(0,1), \quad i = 1, 2, \ldots, n$$

Este es el típico caso emulado por los *generadores de secuencias pseudo–aleatorias* utilizados en la mayoría de los lenguajes de programación (en la presente Tesis se los llamará RND). Es interesante observar particularmente este caso, dado que los modelos prácticos STDEVS serán, en la mayoría de los casos, simulados utilizando funciones RND.

Se provee entonces el siguiente corolario del Teorema 3.1, particularizando las propiedades del modelo STDEVS cuando se utilizan funciones RND en las definiciones de las transiciones de estado.

Corolario 3.1. *Un modelo medible DEVS–RND en el cual $\delta_{int}(s,r)$ depende de n funciones RND (es decir, $r \sim U(0,1)^n$) define un modelo STDEVS equivalente.*

Este corolario no requiere de una prueba, dado que se trata de un caso particular del Teorema 3.1, tomando $R = [0,1]^n$. Aún así, es posible hacer referencia explícita a los componentes del modelo STDEVS resultante.

Procediendo como en el caso general, para cada *conjunto imagen* $G_{s,B} \in \mathcal{G}(s)$, la probabilidad de transicionar desde un estado s a un nuevo estado perteneciendo al conjunto $G_{s,B}$ será:

$$P_{int}(s, G_{s,B}) = P(r \in B)$$

lo cual resulta ser la medida de Lebesgue para el conjunto B.

3.6.2. DEVS y STDEVS

En el caso de que una (o ambas) funciones de transición sea determinista, aún podrá definirse como $\delta(\cdot, r)$, pero de tal manera que resulte independiente de r. Luego, todo el análisis previo se mantiene válido.

Siguiendo este razonamiento, el teorema presentado en esta sección es una forma alternativa de probar que los modelos DEVS deterministas y medibles [8] son un caso particular del caso STDEVS estocástico, con la particularidad de que se elimina la aleatoriedad de la dinámica que rige la transición de estados.

Para finalizar, si se considera que ambas funciones de transición son deterministas (el caso DEVS clásico) y aplicamos el mismo concepto de definirlas como $\delta(\cdot, r)$ independientemente de r, puede derivarse el siguiente corolario:

Corolario 3.2. *Un modelo DEVS determinista y medible siempre define un modelo equivalente STDEVS estocástico.*

3.6.3. Acoplamiento de DEVS y STDEVS

Una característica importante de los modelos STDEVS es que deberían ofrecer una integración transparente con modelos DEVS medibles para conformar

[8] Como en el caso de DEVS–RND, se dirá que un modelo DEVS es medible cuando todas sus funciones también lo sean.

estructuras híbridas. El acoplamiento para modelos DEVS atómicos (junto con su propiedad de clausura bajo acoplamiento) fué definida en [ZKP00], y ahora para modelos atómicos STDEVS en la Sección 3.4.3 y la Sección 3.5.1 de la presente Tesis. Con el objetivo de garantizar conectividad entre ambos tipos de modelos es suficiente probar que la estructura de modelo atómico DEVS medible puede ser siempre conectada a una estructura de modelo atómico STDEVS. Con este fin, se puede hacer uso del Corolario 2, y considerar a cualquier modelo atómico DEVS medible dado como su equivalente STDEVS. Luego, la clausura bajo acoplamiento para STDEVS garantiza que una conexión válida entre ambos tipos de modelos resulta siempre en un nuevo modelo STDEVS acoplado. Esto permite además la composición de modelos por medio de su interconexión jerárquica de modelos DEVS medibles y modelos STDEVS, implicando que un modelo acoplado STDEVS puede ser utilizado, a su vez, dentro de cualquier modelo acoplado más complejo; procedimiento conocido como acoplamiento jerárquico. Este procedimiento puede extenderse tantas veces como se requiera y hasta cualquier número de niveles jerárquicos, permitiendo combinar naturalmente modelos DEVS medibles y STDEVS acorde a las necesidades del usuario.

Cabe resaltar que es suficiente con tener un único modelo atómico STDEVS como parte de una estructura jerárquica y compleja de modelos, para tener inmediatamente la necesidad de contar con una descripción formal de tipo STDEVS para el sistema acoplado completo.

3.6.4. Resumen y Discusión

Se presenta un cierre al cuerpo teórico de STDEVS con una clasificación de los formalismos discutidos hasta aquí: DEVS, DEVS–RND y STDEVS, desde el punto de vista de las principales propiedades que distinguen a los modelos que pertenecen a cada uno de dichos formalismos, como ser la medibilidad y la dimensión de sus conjuntos y funciones.

Además, se relacionará la clasificación ofrecida con aquellos problemas iniciales encontrados en los modelos presentados en la Sección 3.3, para brindar una comprensión integral del contexto teórico en el cual operan.

Cada conjunto en el diagrama de Venn de la Figura 3.2 representa un familia particular de modelos. Los límites entre los conjuntos implican alguna forma de distinción de acuerdo a las capacidades de cada familia de representar los problemas de modelado que se pudieran existir. Los formalismos se componen con los conjuntos acorde a DEVS=$1 \cup 2$, DEVS–RND=DEVS$\cup 3 \cup 4$ y STDEVS=$2 \cup 4 \cup 5$.

La diferencia entre los modelos que pertenecen a $1 \cup 2$ (DEVS clásico) y aquellos que pertenecen a $3 \cup 4$ es que los últimos son capaces de representar comportamiento estocástico (hasta cierto punto). Como se vió en la Sección 3.3.1, las funciones en DEVS clásico no pueden representar formalmente comportamiento estocástico, porque esto conduce a una situación en la que pierden la propiedad de asignar un único valor a cada elemento de un conjunto de entrada dado. Como se muestra en la Figura 3.2, todos los modelos DEVS definen modelos DEVS–RND.

Otra diferencia importante es la que existe entre los modelos que pertenecen a $2 \cup 4$ y aquellos que pertenecen a $1 \cup 3$. Los modelos en $2 \cup 4$ requieren operar sobre conjuntos *medibles*, no así los modelos en $1 \cup 3$. Es por esto que los modelos

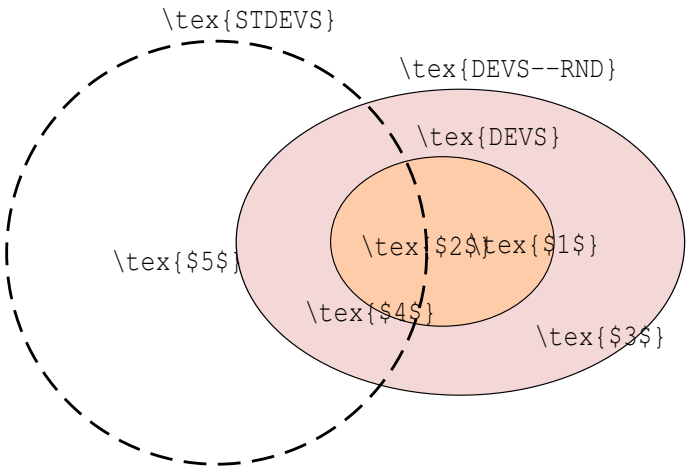


Figura 3.2: Relación entre categorías de modelos estocásticos y deterministas.

en $2 \cup 4$ se denominan *DEVS–RND medibles*, y los modelos en 2 se denominan *DEVS medibles*.

STDEVS fue definido teniendo en cuenta que los procesos estocásticos consistentes deben operar sobre conjuntos medibles. De este modo, STDEVS no puede representar modelos en $1 \cup 3$. Sin embargo, STDEVS *extiende* al dominio de DEVS–RND. La familia de modelos número 5 es exclusiva para la representación de modelos STDEVS.

Estos modelos son diferentes del resto debido a su capacidad de representar comportamiento estocástico sobre espacios (medibles) *infinito–dimensionales*, lo que provee la propiedad de *generalidad* que se requirió en el ejemplo de motivación de la Sección 3.3.2. STDEVS asimismo define modelos DEVS–RND medibles (familia 4) y modelos DEVS medibles (familia 2). La incapacidad de STDEVS para representar modelos en $1 \cup 3$ es en realidad, una ventaja, porque evita que se intente definir con STDEVS modelos inconsistentes (tanto a nivel modelo atómico como modelo acoplado).

Todas las aclaraciones realizadas en los párrafos anteriores pueden ser ilustradas por medio de los ejemplos de motivación discutidos en la Sección 3.3. El modelo M_U de la Sección 3.3.1 pertenece a la familia 4 (es decir, es in modelo DEVS–RND operando sobre espacios medibles). El mismo define también un modelo STDEVS. El modelo descrito en la Sección 3.3.2 pertenece claramente a la familia 5 (es decir, es un modelo STDEVS operando sobre espacios infinito–dimensionales que no pueden representarse por DEVS–RND). El modelo M_V de la Sección 3.3.3 es inconsistente debido a que intenta asignar una medida sobre un conjunto no medible; luego, es del tipo DEVS–RND en la familia 3, pero no define un modelo STDEVS. En la Sección 3.3.4 el modelo M_A es un DEVS determinista de la familia 1 que no define un modelo STDEVS, y M_U es un modelo consistente DEVS–RND de la familia 4 que no define un modelo STDEVS. A su vez el modelo M_A no es en sí mismo inconsistente, pero cuando se lo acopla con M_U el resultado es un modelo inconsistente que se comporta como el modelo M_V de la familia 3, y no define un modelo STDEVS.

Un último concepto acerca de esta categorización de modelos símil–DEVS

en familias es, nuevamente, la distinción central entre modelado y simulación. La mayoría de todos los modelos prácticos DEVS que hayan sido implementados al presente en cualquier simulador pertenecen a la zona $2 \cup 4$. Esto se debe principalmente a la finitud de la capacidad de representación numérica de la plataformas de cómputo conocidas, que inherentemente conducen a una representación *finita y medible* de modelos teóricos quizá más sofisticados, para propósitos de *simulación*.

Además, en la mayoría de los casos prácticos, luego de que un sistema es ***modelado*** matemáticamente como un modelo DEVS–RND medible (zona 4 en la Figura 3.2), aún debe asignársele un algoritmo computacionalmente implementable que aproxime satisfactoriamente algunas propiedades estadísticas deseables del sistema original.

Cuando dicho algoritmo existe, y es elegido, el modelo medible DEVS–RND es finalmente ***simulado*** como un modelo DEVS determinista (zona 2 en la Figura 3.2) que utiliza secuencias numéricas pseudo–aleatorias (es decir, secuencias deterministas estrictamente hablando) para evaluar las transiciones de estado.

Aún así, como se demostró a lo largo del presente capítulo, surgen cuestiones importantes cuando se trata con las capacidades y limitaciones teóricas de los formalismos de *modelado* para describir fenómenos estocásticos, *sin limitar* el análisis a los subconjuntos de aquellos casos prácticos implementables computacionalmente.

Estas cuestiones se hacen evidentes y necesarias cuando se incorpora la teoría de Espacios de Probabilidad al formalismo DEVS.

3.7. Caso de Estudio

En esta sección se desarrollarán dos casos de estudio para ilustrar el uso de STDEVS en el modelado de sistemas estocásticos y su relación con el proceso de simulación subsiguiente. Primero se presenta un modelo de un sistema balanceador de carga. Luego, se propone un modelo híbrido para un sistema de control en red. Utilizando la teoría presentada en este capítulo se verá que la representación práctica de DEVS–RND medible para la representación de procesos aleatorios son consistentes con sus representaciones STDEVS, en términos de espacios de probabilidad.

3.7.1. Preliminares acerca del proceso de modelado estocástico y simulación estocástica con DEVS

Como se discutió antes, y particularmente en la Sección3.6.4), la distinción entre modelado y simulación es vital para garantizar un proceso correcto de estudio de sistemas estocásticos.

Efectivamente, ya se estableció que existen modelos DEVS simulables en forma práctica pero que conducen a resultados completamente incorrectos desde el punto de vista de las probabilidades (por ejemplo, aquellos que manejan conjuntos no medibles, que pueden definirse con DEVS); y también que existen modelos matemáticos estocásticos completamente válidos pero que nunca podrían ser simulados exactamente con las computadoras existentes (por ejemplo, aquellos que manejan espacios infinito–dimensionales, que pueden definirse con STDEVS). También se estableció que STDEVS es un marco formal que

permite un modelado matemático, consistente y genérico, basado en una teoría robusta: la teoría de Espacios de Probabilidad.

Ahora bien, cuando se trata de casos de estudio prácticos para STDEVS, es importante concentrase en los pasos desde el modelado hasta la simulación que conducen desde un modelo estocástico formalmente robusto hasta una pieza de código ejecutable en una computadora. En la Figura 3.3 se ofrece un diagrama de un proceso paso a paso involucrando el uso de STDEVS. El proceso comprende tres actividades y dos transiciones.

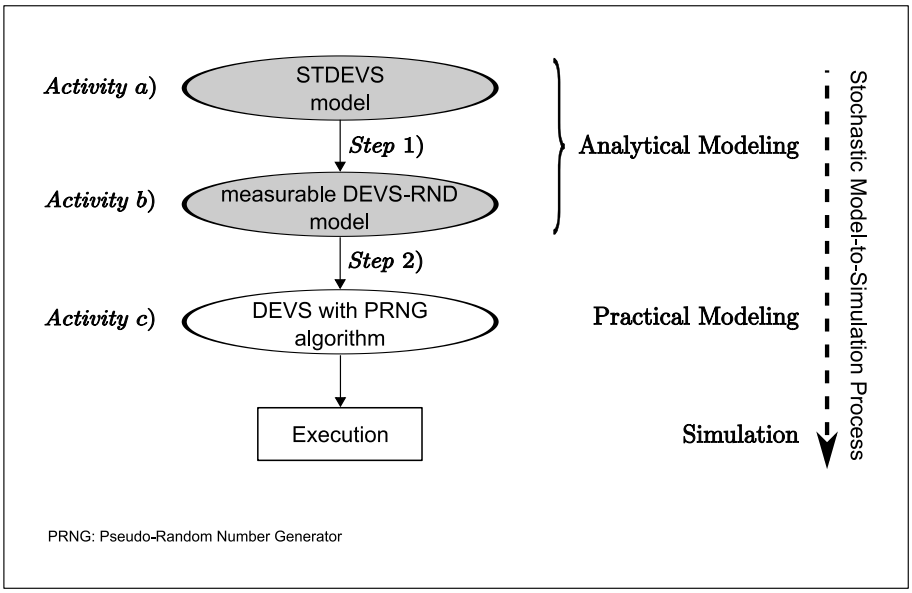


Figura 3.3: Proceso de Modelado–hasta–Simulación estocásticos. Actividades y Transiciones.

Las actividades representan la acción de obtener una representación basada en DEVS de un sistema estocástico dado. El proceso puede comenzar directamente en cualquier *Actividad*. Cuando la información de entrada para una Actividad es únicamente el sistema estocástico del mundo real a ser modelado, no se requiere de ningún *Paso* de transición entre modelos. Cuando una nueva Actividad se inicia proviniendo de una Actividad previa que ha sido completada, cada Paso transicional involucra la adaptación del modelo estocástico de una forma de representación a otra (es decir, un procedimiento de transformación de modelos).

En esta tesis se postulará que *el hecho de comenzar el proceso de modelado estocástico en cualquier otro punto que no sea la* Actividad *a*) *incrementa el riesgo metodológico de obtener un modelo incorrecto o impreciso*. Sin embargo, en ciertos casos, el riesgo puede ser suficientemente comprendido y/o considerado lo suficientemente bajo como para proceder comenzando por cualquier otro punto de inicio en el proceso sugerido.

Por ejemplo, comenzando directamente por la *Actividad c*) podría, como no, conducir a la reproducción de un modelo válido y consistente, y su precisión dependerá de la experiencia teórica previa que tenga el programador.

Por supuesto que existen distribuciones probabilísticas prácticas vastamen-

te utilizadas y muy conocidas, que se encuentran directamente disponibles en forma de descripciones algorítmicas [Knu97] o incluso en forma de bibliotecas de dominio público [Ins09] reutilizables en los lenguajes de programación más populares.

Sin embargo, se hace obvio que estos "atajos prácticos" (cuando son correctos) están confinados a un subconjunto muy particular de modelos estocásticos: aquellos que *a*) son representables por descripciones DEVS–RND medibles y *b*) preservan satisfactoriamente algunas propiedades estocásticas cuando son implementados en un simulador DEVS usando generadores de números pseudo–aleatorios.

Evidentemente, la existencia de la mencionada vasta familia de algoritmos prácticos disponibles en la literatura no proveen ninguna generalización acerca de su descripción matemática, y no es suficiente para sustituir a un formalismo genérico de modelado estocástico.

Por el contrario STDEVS está más cerca de la descripción analítica de los modelos que de su implementación práctica, y provee una herramienta matemática para garantizar la correctitud de cualquier modelo estocástico DEVS posible en el marco de la teoría de Espacios de Probabilidad, incluyendo aquellos que se implementan en la práctica diaria por parte de la comunidad DEVS.

Dicho de otro modo, cada posible modelo *DEVS medible* o *DEVS–RND medible* simulables ejecutándose en una computadora digital, es una implementación práctica de un modelo STDEVS. En este punto, queda claro que no es posible asegurar que con el solo hecho de generar números al azar en las funciones de transición de algún simulador DEVS se obtendrán garantías acerca de la medibilidad de los modelos DEVS–RND equivalentes. Consecuentemente, no puede asegurarse que exista su representación como un modelo STDEVS, por lo cual no puede realizarse ninguna afirmación acerca de su consistencia con la teoría de Espacios de Probabilidad.

Por lo expuesto hasta aquí es evidente que tiene sentido, en cambio, trabajar del modo sugerido en la Figura 3.3. Es decir, partiendo de la especificación del modelo STDEVS, bajando hacia la especificación del modelo DEVS–RND, y finalmente bajando hasta la implementación en una computadora digital usando generadores de números pseudo–aleatorios. Si este proceso de modelado–hasta–simulación es exitoso, queda garantizado que el código ejecutable cumplirá con los requerimientos teóricos para representar un modelo estocástico consistente.

Nota: Como se verá inmediatamente, en la práctica, las cosas son mucho más sencillas cuando se conoce de antemano que se está modelando alguna distribución probabilística clásica, que es bien sabido operan sobre espacios de probabilidad bien definidos. En situaciones como estas (quizá, las mas frecuentes) se puede recurrir al uso del Teorema 3.1 y argumentar que las descripciones tipo *DEVS–RND medible* de esos modelos ***son también modelos STDEVS***. Esta situación permite evitar la *Actividad a*) y el *Paso* 1) en el proceso de modelado [9] lo cual es una ventaja en términos de reducción de esfuerzos. Desde la *Actividad b*) hacia abajo en la Figura 3.3 el proceso no difiere del caso típico aplicado históricamente en el modelado y simulación de sistemas estocásticos con DEVS.

[9] Con esta decisión, se acepta el riesgo involucrado, porque a) se conoce que el riesgo existe y b) se posee una cantidad suficiente de conocimiento previo sobre el problema como para tener confianza en que dicho riesgo es extremadamente bajo.

El objetivo de los ejemplos siguientes será mostrar en acción el proceso completo de modelado–hasta–simulación utilizando explícitamente modelos STDEVS en ciertos casos, o recurriendo al Teorema 3.1 en otros, completando los procedimientos típicos de modelado, simulación y análisis de resultados con DEVS. Debido a que se pretende alcanzar situaciones simulables, los ejemplos seleccionados no presentarán ninguna dificultad teórica para completar el proceso exitosamente.

3.7.2. Sistema Balanceador de Carga

El ejemplo *Load Balancing Model* (LBM) presentado en esta sección es una simplificación de un sistema de cómputo que procesa una carga ofrecida de tareas. Este ejemplo muestra un sistema en el cual la dinámica depende completamente de experimentos aleatorios.

El LBM se compone de los siguientes modelos atómicos: *Load Generator* (LG), *Weighted Balancer* (WB) y dos *Servers* (S1,S2) sin capacidad de encolamiento (es decir, las tareas que llegan a un Server ocupado, son descartadas). El conjunto {WB,S1,S2} compone el subsistema *Cluster* (CL), un modelo acoplado.

Como se hizo antes, las funciones de transición se expresarán en términos de $r \sim U(0,1)$, concretamente $\delta_{int}(\cdot) = \delta_{int}(s,r)$ y $\delta_{ext}(\cdot) = \delta_{ext}(s,e,x,r)$.

Load Generator (LG)

Este modelo genera un número de tareas por unidad de tiempo acorde a una distribución aleatoria de Poisson siendo d_r la *tasa esperada de partida* de nuevas tareas. Puede probarse que los tiempos de inter–generación σ_k entre las tareas k y $k+1$ están exponencialmente distribuidas de acuerdo a $P(\sigma_k \leq t) = 1 - e^{-at}$ en donde $a = d_r$ y $1/a$ es el valor medio esperado. Se asumirá que el modelo LG genera un único tipo de tareas ($task_1$) que se emiten por el único puerto de salida (out_1). El modelo LG no posee entradas, de modo que sólo son posibles las transiciones internas. La definición STDEVS para LG es:

$$M_{ST}^{LG} = (X, Y, S, \mathcal{G}_{int}, \mathcal{G}_{ext}, P_{int}, P_{ext}, \lambda, ta)$$

con los componentes deterministas:

- $X = \oslash, Y = \{(task_1, out_1)\}$
- $S = \Re_0^+$
- $\lambda(s) = (task_1, out_1)$
- $ta(s) = s$

y funciones estocásticas:

- $\mathcal{G}_{int}(s) = \{A_t \mid t \geq 0\}$, $A_t = [0, t)$
- $P_{int}(s, G) = P_{int}(s, A_t) = 1 - e^{-at}$, $G \in \mathcal{G}_{int}$

Como puede observarse, la descripción estocástica para el tiempo de inter–generación de tareas está asociado directamente con la función P_{int} por medio

de su correspondiente función de distribución acumulada *cdf* (cumulative probability function). Debido a que únicamente son posibles las transiciones internas, no es necesario definir $\mathcal{G}_{ext}, P_{ext}$.

Para implementar este modelo STDEVS en una computadora digital, la descripción probabilística debe ser traducida en un algoritmo que sea ejecutado dentro del código de la transición interna que representa la función DEVS asociada $\delta_{int}(\cdot)$. Esto es, se pasa desde la *Actividad a*) a la *Actividad b*) completando el *Paso* 2) en el proceso de la Figura 3.3.

Para cumplir esto, y de acuerdo a las definiciones previas, se define:

$$\delta_{int}(s, r) = -(1/a)log(r)$$

donde por medio del método de transformación inversa, se obtiene una distribución de tipo exponencial haciendo uso de una variable distribuida uniformemente $r \sim U(0, 1)$ (disponible como una función RND() en la mayoría de los lenguajes de programación).

Como consecuencia, la especificación equivalente DEVS–RND medible para LG es:

$$M_D^{LG} = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta)$$

donde:

$$\begin{cases} X = \oslash \\ Y = \{(task_1, out_1)\} \\ S = \Re_0^+ \\ \delta_{int}(s, r) = -(1/a)log(r) \\ \delta_{ext}(s, e, x, r) = s \\ \lambda(s) = (task_1, out_1) \\ ta(s) = s \end{cases}$$

En este componente, el próximo tiempo de partida calculado aleatoriamente se almacena en el estado s (en este caso, un número real) el cual es utilizado luego por la función de avance de tiempo $ta(s) = s$ poniendo a LG "inactivo" durante ese período [10].

Notar que si se adoptase una n-tupa real para el estado s, se estaría en condiciones de aplicar distintas distribuciones aleatorias independientes para decidir el próximo valor del estado para cada uno de los n elementos de s, luego de cada transición de estado. Entonces, con sólo incluir el avance de tiempo σ como uno de los n elementos componentes de s y luego utilizar σ para evaluar $ta(s)$, se está en condiciones de aplicar cualquier distribución estocástica para decidir el tiempo de vida de cada estado próximo inmediato. Si m de los n componentes de s ($m \leq n$) se debiesen definir en base a generadores aleatorios, entonces podría usarse un vector aleatorio $r \sim U(0, 1)^m$ de dimensión m.

Nota: De aquí en más, se construirán los modelos del presente ejemplo directamente como modelos DEVS–RND. Esto es, se hará uso del Teorema 3.1 y se hará referencia únicamente a la forma DEVS–RND de los componentes con

[10] Un razonamiento similar puede ser aplicado para el resto de los componentes, en donde las variables de estado son utilizadas con fines de almacenamiento de información (memoria)

aleatoriedad (conteniendo funciones RND en los algoritmos de sus funciones de transición de estado). Esto es posible porque los modelos utilizados representarán distribuciones clásicas de probabilidad pertenecientes a espacios de probabilidad finito–dimensionales bien definidos, produciendo modelos DEVS–RND medibles. Este es un enfoque de modelado más sencillo que el de definir la estructura STDEVS (como se hizo antes para LG), porque en DEVS–RND no es necesario definir las funciones estocásticas $\mathcal{G}_{int}, \mathcal{G}_{ext}, P_{int}, P_{ext}$. Aun así, por medio del Teorema 3.1, el modelo STDEVS equivalente podrá siempre ser obtenido desde un modelo DEVS–RND *medible*, construyendo la estructura STDEVS siguiendo razonamientos similares a los aplicados para LG.

Weighted Balancer (WB)

El componente WB envía las tareas entrantes en su puerto de entrada (Port: inp_1) hacia los puertos de salida out_1 y out_2 basándose en un factor de balanceo $b_f \in [0,1]$ que determina la relación de peso entre ambos puertos. Para $b_f = 0{,}5$ ambos puertos de salida tienen el mismo peso, y luego la carga de salida será balanceada de forma equiprobable. Para $b_f > 0{,}5$ se privilegia out_1 y para $b_f < 0{,}5$ se privilegia out_2), en forma lineal. Las tareas aceptadas pertenecen a un conjunto $T = \{task_1, ..., task_m\}$ con m posibles tareas diferentes.

La definición DEVS–RND medible M_D^{WB} para WB es:

$$M_D^{WB} = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta)$$

con:

- $X = T \times \{inp_1\}$, $Y = T \times \{out_1, out_2\}$
- $S = T \times \{out_1, out_2\} \times \Re_0^+$
- $\lambda(w, p, \sigma) = (w, p)$
- $ta(w, p, \sigma) = \sigma$

El estado es un triplete $s = (w, p, \sigma)$, donde w representa la última tarea recibida, p es el puerto de salida por donde se enruta la tarea y σ es el avance de tiempo. Para este ejemplo $T = \{task_1\}$. Luego, al recibir un elemento externo (x_v, x_p) el nuevo estado debe ser evaluado por:

$$\delta_{ext}((w, p, \sigma), e, (x_v, x_p), r) = (x_v, \tilde{p}, 0)$$

con

$$\tilde{p} = \begin{cases} out_1 & \text{if } r < b_f, \\ out_2 & \text{en cualquier otro caso.} \end{cases}$$

Finalmente, la transición interna será:

$$\delta_{int}((w, p, \sigma), r) = (w, p, \infty)$$

en este caso, independiente de r.

Server1 y Server2 (S1,S2)

Los servidores S1 y S2 son componentes que reciben las tareas entregadas por el balanceador WB. Los servidores procesan cada tarea recibida con un tiempo de servicio s_t. Una vez procesada, la tarea es enviada hacia un sumidero (sink), en donde es reconocida como una "tarea completada". El tiempo de servicio variable s_t está distribuido exponencialmente con $P(s_t \leq t) = 1 - e^{-bt}$, y su valor medio esperado es $1/b$. No existe política de encolado ni de reordenamiento por prioridades en los servidores. Si una nueva tarea llega a un servidor ocupado, la misma es descartada.

La definición DEVS–RND medible $M_D^{S_n}$ con $n = 1, 2$ para los servidores S_n es:

$$M_D^{S_n} = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta)$$

donde:

- $X = T \times \{inp_1\}$, $Y = T \times \{out_1\}$
- $S = T \times \{false, true\} \times \Re_0^+$
- $\lambda(w, busy, \sigma) = (w)$
- $ta(w, busy, \sigma) = \sigma$

El estado es un triplete $s = (w, busy, \sigma)$, donde w representa la última tarea recibida, $busy$ representa la condición lógica de ocupación actual del server (si $busy = true$ el servidor está procesando una tarea, y si $busy = false$ el servidor está libre) y σ es el tiempo restante hasta el próximo evento agendado. Para este ejemplo se tomará $T = \{task_1\}$, un único puerto de entrada y un único punto de salida. Luego de recibir un evento (x_v, x_p) el nuevo estado será evaluado acorde a:

$$\delta_{ext}((w, busy, \sigma), e, (x_v, x_p), r) = (\tilde{w}, true, \tilde{\sigma})$$

con

$$\begin{cases} \tilde{w} = x_v, \tilde{\sigma} = -(1/b)log(r) & \text{si } no(busy), \\ \tilde{w} = w, \tilde{\sigma} = \sigma - e & \text{si } busy. \end{cases}$$

with $r \sim U(0, 1)$. y la transición interna será completada como:

$$\delta_{int}((w, busy, \sigma), r) = (w, false, \infty)$$

independiente de r.

Modelo Acoplado LBM

Como se discutió antes, este modelo pretende mostrar un escenario en donde las variables aleatorias afectan a todas sus partes componentes. Se tiene un proceso de Poisson para la generación de tareas, un proceso Uniforme (con una posterior tendencia o desvío determinista) afectando al balanceador entre dos servidores y un proceso Exponencial representando los tiempos de servicio en cada server. Sin embargo, la implementación siempre recae en el uso de una variable auxiliar distribuida uniformemente: $r \sim U(0, 1)$.

En la Figura 3.4 se muestra la topología del modelo que se utilizará en adelante, con sus principales parámetros y magnitudes de tráfico derivadas.

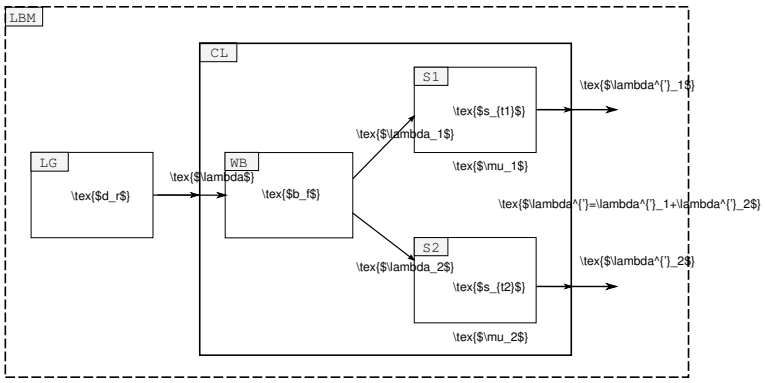


Figura 3.4: Topología del Load Balancer Model (LBM).

Simulaciones

Basándose en la especificación DEVS–RND medible de los componentes y sus interconexiones, se construye el modelo completo en la herramienta de simulación (PowerDEVS [KLP03]). Esto implica completar el *Paso* 2) y la *Actividad c*) en el proceso descrito en Figura 3.3. No se entrará en mayor detalle en esta parte dado que no aporta ningún concepto nuevo ni complejidad interesante para esta Tesis. Se hará referencia al modelo como el *modelo STDEVS*, dado que es sabido que al seguir el proceso de Modelado–hasta–Simulación de sistemas estocásticos propuesto, se obtienen siempre instancias particulares de modelos que son *explicados* por la teoría genérica STDEVS.

Se ejecutaron varias simulaciones con el sistema en diversos puntos de operación. Para validar al modelo STDEVS, se comparó los resultados de simulación contra resultados analíticos de la teoría de colas [Kle75].

Un servidor único y sin capacidad de encolamiento puede describirse por un sistema $M/M/m/m$ con $m = 1$. Esta descripción asume tiempos de arribo y tiempos de servicio distribuidos exponencialmente (lo que concuerda con el ejemplo tratado). Para el i–ésimo servidor se tienen los parámetros λ_i (*tasa de arribo*) y μ_i (*tasa de servicio*). La *intensidad de tráfico* se define como:

$$\rho_i = \lambda_i / \mu_i \tag{3.9}$$

Debido a que el sistema sólo puede almacenar un único elemento (el que está siendo procesado) existe una probabilidad de perder tareas, que nunca recibirán servicio (se eliminan del sistema), y que se denota P_{loss_i} (*probabilidad de pérdida*). Su relación con la intensidad de tráfico está dada por la *fórmula de pérdida de Erlang* [Kle75] en su forma más simple para un único servidor:

$$P_{loss_i} = \rho_i / (1 + \rho_i) \tag{3.10}$$

El i–ésimo servidor "verá" en su puerto de entrada una *tasa de arribo efectiva*:

$$\lambda_i^{'} = \lambda_i (1 - P_{loss_i}) \tag{3.11}$$

la cual, bajo condiciones de estabilidad [11] es igual a la *tasa efectiva de transmisión* (o "throughput") del servidor en su puerto de salida. En el ejemplo LBM se tiene $i = 1, 2$ para los servers en el submodelo Cluster (CL). La *tasa efectiva total de transmisión* $\lambda^{'}$ del sistema LBM debe ser $\lambda^{'} = \lambda_1^{'} + \lambda_2^{'}$ siendo entonces una función de la *tasa total de arribo* λ al sistema y de las intensidades de tráfico ρ_1, ρ_2 en los servidores.

Estas magnitudes se calculan usando los siguientes parámetros de modelo: *tasa promedio de partida* d_r desde LG (o "departure rate") en *Tareas/segundo*), *factor de balanceo* b_f en WB (o "balancing factor"), *tiempo promedio de servicio* s_{t1}, s_{t2} en S1 y S2 respectivamente (o "mean service time") en *segundos*, como sigue:

$$\begin{aligned} & \lambda = d_r \\ \mu_1 = 1/s_{t1} \qquad & \lambda_1 = b_f \lambda \\ \mu_2 = 1/s_{t2} \qquad & \lambda_2 = (1 - b_f)\lambda \end{aligned} \tag{3.12}$$

Ahora, con (3.9) y (3.12) en (3.10) se derivan las probabilidades de pérdida internas:

$$P_{loss_1} = \frac{b_f d_r s_{t1}}{1 + b_f d_r s_{t1}}, P_{loss_2} = \frac{(1 - b_f) d_r s_{t2}}{1 + (1 - b_f) d_r s_{t2}} \tag{3.13}$$

Finalmente, se desea expresar la *tasa efectiva total* del sistema $\lambda^{'}$ (o "total system throughput") en términos de una *probabilidad global de pérdida* (o "global loss probability") P_{loss} como se hizo para cada Server en forma individual. Así, considerando (3.11) y (3.13) se obtiene:

$$\begin{aligned} P_{loss} &= b_f P_{loss_1} + (1 - b_f) P_{loss_2} \\ \lambda^{'} &= \lambda(1 - P_{loss}) \end{aligned} \tag{3.14}$$

Con las Ecuaciones (3.14) se caracteriza completamente al sistema en términos de carga ofrecida, probabilidad de pérdida y tasa efectiva. La Figura 3.5 muestra las curvas teóricas de $P_{loss}, P_{loss_1}, P_{loss_2}$ y $\lambda^{'}$ como funciones de b_f en el *Escenario de Test 1* elegido como $TS_1 = \{d_r = 10, b_f \in [0, 1], s_{t1} = 0{,}2, s_{t2} = 0{,}2\}$. En la misma figura se grafican los resultados de simulación para el modelo STDEVS LBM parametrizado acorde al escenario de test TS_1, en un conjunto de puntos de operación ilustrativos que barren b_f entre 0 y 1.

Puede observarse que los resultados de simulación se ajustan muy bien a las curvas teóricas, para 15 repeticiones de los experimentos en cada punto de operación simulado. Cada valor de los puntos de operación graficados se obtuvo procesando el archivo de *historial de eventos* (o "event log") generado en cada simulación, calculando la variable *task rate* [12] para obtener $\lambda^{'sim}$ y $P_{loss_i}^{sim} = 1 - (\lambda_i^{'sim}/\lambda_i^{sim})$. Se verificaron las propiedades estadísticas de las variables aleatorias producidas por los modelos atómicos, acordando con las esperadas: distribución uniforme para b_f, distribución discreta de Poisson para λ y distribución exponencial para s_{t1} y s_{t2}.

[11] En sistemas con pérdida, la *intensidad efectiva de tráfico* $\rho_i^{'} = \lambda_i^{'}/\mu_i$ es siempre $\rho_i^{'} < 1$, de modo que la típica condición de estabilidad: $\lambda_i/\mu_i < 1$ no necesita invocarse. Los sistemas con buffers finitos son siempre estables ya que los elementos entrantes se descartan (nunca llegan a formar parte del sistema) cuando el número de elementos en el sistema excede su capacidad.

[12] A general λ_k^{sim} task rate at an arbitrary observation place k is: $\lambda_k^{sim} = NumberOfTasksLogged_k/TotalsimulationTime$.

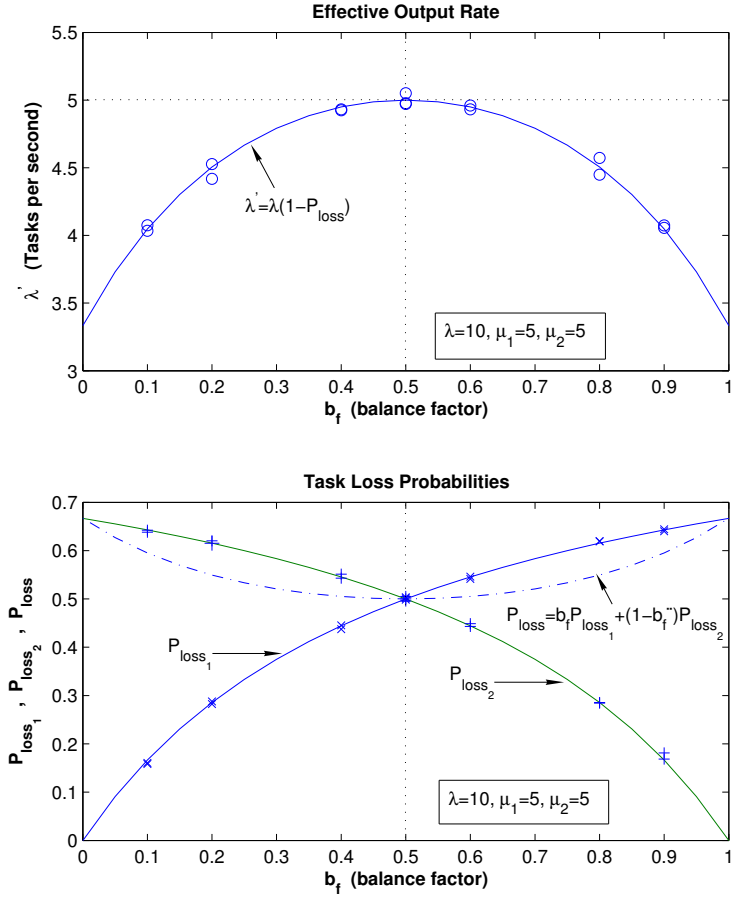


Figura 3.5: Resultados de simulación (puntos) vs. Curvas teóricas (líneas). Escenario de Test 1: $d_r = 10, b_f \in [0,1], s_{t1} = 0{,}2, s_{t2} = 0{,}2$

3.7.3. Sistema de Control en Red

En este ejemplo, se muestra otra representación DEVS práctica de procesos aleatorios, consistentes con sus especificaciones STDEVS, en un sistema que incluye modelado híbrido y Teoría de Control. Presentamos el modelo híbrido de un *Sistema de Control en Red* (o "Networked Control System", NCS) [NBW98, GH01] cuyos componentes responden a señales de tiempo continuo, tiempo discreto y eventos discretos. Los NCS son Sistemas de Control cuyos lazos de control se cierran por medio de redes de tiempo real, en donde la información de las señales principales se distribuyen entre los componentes del sistema por medio de las redes subyacentes.

El diagrama de bloques conceptual del modelo se muestra en la Figura 3.6. El sistema consiste en una planta LTI (Linear Time-Invariant) modelada como un sistema SISO (Single-Input Single-Output) en la cual la señal de Salida $y'(t)$

debe mantenerse en cierto valor dado por la Referencia de Entrada. Para lograr esto, se implementa un sistema de control realimentado, el cual utiliza una Red de Control compartida para comunicar la información sensada a la Salida hacia el módulo de control.

El sistema es perturbado por dos procesos estocásticos. En primer lugar, la señal continua de salida de la planta (a ser controlada) se ve perturbada por un proceso aleatorio continuo modelado como Ruido Blanco Aditivo Gaussiano (AWGN, por *Additive White Gaussian Noise*), en el cual se concentra la representación de todas las perturbaciones del sistema que deben ser compensadas por el controlador. Este tipo de ruido posee densidad espectral de potencia constante, una distribución Gaussiana de amplitud, y se considera que afecta de manera lineal a la señal perturbada.

Por su parte, un proceso aleatorio con distribución Uniforme modela los retardos inducidos por la red que afectan a los paquetes de datos que transportan la información desde el sensor hasta el controlador.

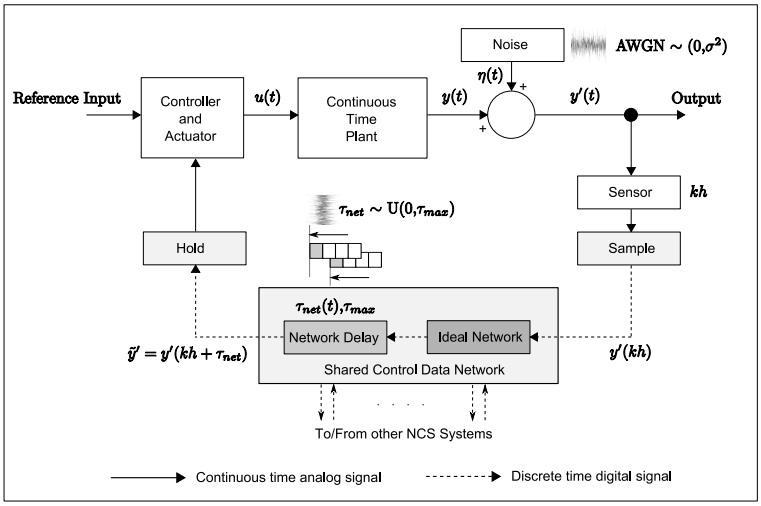


Figura 3.6: Topología del ejemplo del sistema híbrido NCS (Networked Control System)

La parte continua del modelo consiste en la Referencia de Entrada, el Controlador, el Actuador, la Planta, la perturbación Ruido Blanco Aditivo Gaussiano (AWGN) sumada a la señal de salida de la planta (antes de ser medida) y el Sensor.

El subsistema Controlador/Actuador es un componente a tiempo continuo y a eventos discretos, que calcula señales de control y actúa sobre las entradas de la planta cada vez que recibe nueva información de sensado proveniente de la red compartida.

La parte a tiempo discreto del modelo se compone de un Sensor, la Red de Control, y los respectivos conversores analógico–digital (Sample) y digital–analógico (Hold).

A través de este lazo, la señal ruidosa de la planta es sensada y realimentada de modo periódico en base a un reloj digital. Esta realimentación es afectada por un tiempo extra de retardo, aleatorio y acotado superiormente. Este retar-

do aleatorio sintetiza la superposición resultante de la concurrencia de todas las potenciales fuentes de demora presentes en la red (por ejemplo, efectos de encolamiento, acceso al medio físico, tiempo de procesamiento de paquetes, políticas de priorización, etc.). Estos efectos se producen debido a que la Red de Control es un recurso compartido y limitado (en ancho de banda) por el cual compiten otros NCS.

El lazo cerrado se compone del Sensor, la Red de Control y el camino Controlador/Actuador, es decir, un lazo híbrido que combina señales a tiempo continuo y tiempo discreto, y componentes manejados por un reloj (síncronos) con otros que evolucionan a eventos discretos (asíncronos).

Se definirá al sistema completo utilizando STDEVS y se implementará en el simulador basado en eventos discretos PowerDEVS. También se implementará el sistema en el conocido simulador Matlab-Simulink basado en tiempo discreto. Se definirá una función de Costo de Control y se la estudiará bajo distintas condiciones de la Red de Control ejecutando simulaciones con ambas herramientas y se compararán los resultados.

Especificación del Sistema NCS

El valor de la Referencia de Entrada a ser copiado por la Salida de la Planta se define como $Ref = 1$. La Planta a tiempo continuo se describe por medio de su Función Transferencia –una representación matemática en términos de frecuencias– de la relación entre la Entrada y la Salida de un sistema LTI. La función transferencia $G_P(s) = Y(s)/U(s)$ en el dominio de Laplace es una relación lineal entre la transformada de Laplace de la entrada $U(s) = \mathcal{L}(u(t))$ contra la transformada de Laplace de la salida $Y(s) = \mathcal{L}(y(t))$, siendo s la frecuencia compleja del sistema ($s = jw = j2\pi f$).

Se define a la función transferencia de la planta como $G_P(s) = \frac{1}{s^2+0{,}8s}$, la cual representa un sistema inestable a lazo abierto (es decir, si lazo de realimentación entre la entrada y la salida) que se estabiliza bajo una condición de realimentación unitaria a lazo cerrado (que será el caso del ejemplo que se estudiará) A la salida de la planta, el ruido AWGN $\eta(t)$ tiene valor medio 0 y varianza $v_\eta = 0{,}001$. El período de muestreo síncrono para la parte de tiempo discreto será h=1 segundo, y el retardo aleatorio estará uniformemente distribuido acorde a $\tau_{net}(s) \sim U(0, \tau_{max})$.

También, el retardo está limitado con $\tau_{max} < h$, de modo que no existe posibilidad de encolamiento en el componente que modela el retardo de red aleatorio.

Modelo STDEVS del NCS

En esta sección se presenta una implementación del sistema NCS en PowerDEVS. En la Figura 3.7 se muestra el diagrama de bloques del modelo STDEVS NCS, incluyendo una rama (en la parte inferior) que calcula la función de costo utilizada para análisis de resultados (ver detalles en la Sección 3.7.3). Se toma como referencia la topología conceptual del sistema NCS en la Figura 3.6, y para cada parte de la topología se describirá su correspondencia con su componente STDEVS equivalente en el diagrama de bloques de la Figura 3.7.

La Entrada de Referencia es provista por el componente *Reference Generator* con un valor constante igual a 1. Los subsistemas Controlador y Actuador son

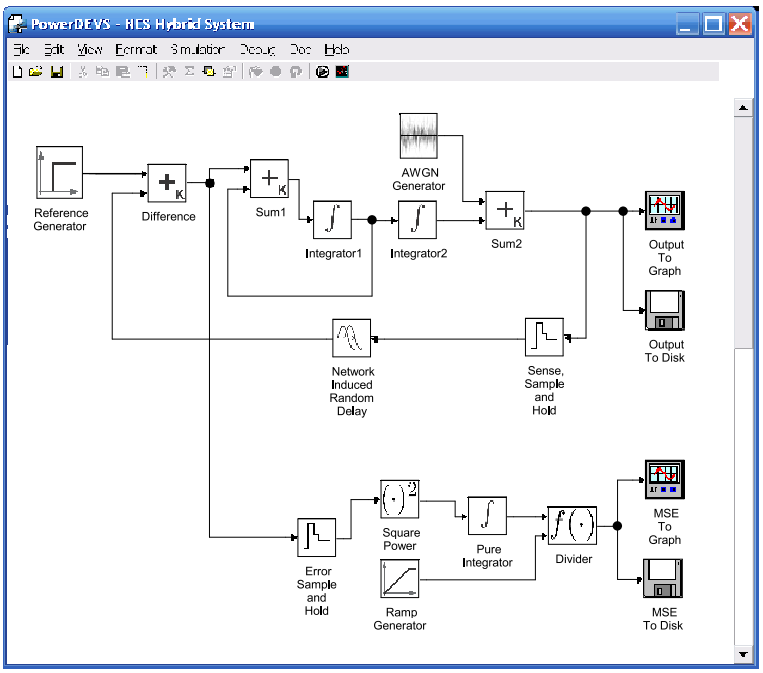


Figura 3.7: Modelo del NCS en PowerDEVS.

resueltos por el componente *Difference*, dado que se trata simplemente de un lazo de control unitario. La Planta de tiempo continuo consta de los componentes *Integrator1*, *Integrator2* y *Sum1*; con el componente aditivo Noise afectando su señal de Salida (provisto por los componentes *AWGN Generator* y *Sum2*).

Las funcionalidades de Sensor, Sample y Hold se resuelven por un único componente *Sense, Sample and Hold*. Finalmente, el componente Shared Control Data Network se modela con un único componente denominado *Network Induced Random Delay*.

Son de especial interés los integradores a tiempo continuo y los generadores estocásticos AWGN y Random Delay. En el caso de los integradores, el método de integración QSS [CK06] se utilizó para aproximar numéricamente el subsistema de tiempo continuo [13]. En la sección que sigue se discute la definición de los componentes estocásticos usando sus especificaciones formales STDEVS.

El resto de los componentes definidos como especificaciones DEVS deterministas se omiten por razones de brevedad, y pueden encontrarse en [CK08a].

Especificación formal para componentes estocásticos

Los componentes que se describirán aquí son los generadores *AWGN Generator* (AWGN) y *Network Induced Random Delay* (NIRD). Como se hizo en

[13] Los métodos QSS discretizan los sistemas continuos mediante la cuantización de las variables de estado, en lugar de discretizar el tiempo. El sistema resultante es equivalente a un modelo DEVS.

el primer ejemplo (Sección 3.7.2) para especificaciones STDEVS, se acudirá al Teorema 3.1 y se hará referencia únicamente a la forma DEVS–RND de los componentes que tienen su comportamiento estocástico expresado por funciones RND() en sus funciones de transición (representadas aquí por la variable aleatoria r). Nuevamente, esto es posible debido q que se conoce de antemano que los modelos estocásticos en cuestión están definidos sobre espacios de probabilidad finito–dimensionales, resultando en modelos DEVS–RND medibles. Esto simplifica el proceso de modelado evitando tener que definir las funciones estocásticas $\mathcal{G}_{int}, \mathcal{G}_{ext}, P_{int}, P_{ext}$.

Sin embargo, por razones de completitud, se mostrará la formulación STDEVS del componente AWGN, mientras que en cambio se hará uso del Teorema 3.1 para el componente NIRD.

Generador AWGN: El componente AWGN no tiene entradas y tiene un puerto de salida por medio del cual entrega números reales con distribución Gaussiana representando el nivel de la señal de ruido $\eta(t)$. La señal $\eta(t)$ debe obtenerse de una distribución continua, describiendo la probabilidad $P(\eta \leq \varphi)$, donde φ es un valor real que representa todas los posibles niveles de ruido, con $-\infty < \varphi < \infty$. Esta descripción es:

$$P(\eta \leq \varphi) = \Phi_{\mu,v}(\varphi)$$
$$\Phi_{\mu,v}(\varphi) = \frac{1}{\sqrt{v}\sqrt{2\pi}} \int_{-\infty}^{\varphi} e^{-\left(\frac{(u-\mu)^2}{2v}\right)} du \ , \ \varphi \in \Re$$

donde $\Phi_{\mu,v}$ es la función de distribución acumulada *cdf* de la señal, siendo μ y v el valor medio esperado y la varianza, respectivamente. Con esta información, se puede empezar a definir los componentes estocásticos del modelo STDEVS para el generador AWGN.

- $\mathcal{G}_{int}(s) = \{B_\varphi \mid -\infty < \varphi < \infty\}$, with $B_\varphi = (-\infty, \varphi]$
- $P_{int}(s, G) = P_{int}(s, B_\varphi) = \Phi_{\mu,v}(\varphi)$, $G \in \mathcal{G}_{int}$

Como puede verse, la descripción está asociada directamente con la función P_{int} por medio de la correspondiente *cdf*. Luego, la definición STDEVS para AWGN:

$$M_{ST}^{AWGN} = (X, Y, S, \mathcal{G}_{int}, \mathcal{G}_{ext}, P_{int}, P_{ext}, \lambda, ta)$$

puede completarse definiendo el estado $s = (\eta, \sigma)$ (donde η representa el nivel de ruido y σ es el avance de tiempo), y los componentes deterministas:

- $X = \oslash, Y = \Re \times \{out_1\}$
- $S = \Re \times \Re_0^+$
- $\lambda(\eta, \sigma) = (\eta, out_1)$
- $ta(\eta, \sigma) = \sigma$

Dado que en AWGN solo serán posibles las transiciones internas, no es necesario definir $\mathcal{G}_{ext}, P_{ext}$.

Ahora, se avanzará a la *Activity b*) del proceso, y se elegirá un método matemático que permita obtener sucesivos valores η distribuidos de forma gaussiana por medio de la manipulación de resultados de experimentos aleatorios con un generador RND() de distribución uniforme.

Es decir, se pretende encontrar la formulación de tipo DEVS–RND medible del modelo STDEVS AWGN construido en *Actividad a*)

Se denotará el valor medio esperado como μ_η y la varianza de ruido como v_η y se los modelará como parámetros externos.

El modelo puede definirse según su forma DEVS–RND medible como sigue:

$$M_D^{AWGN} = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta)$$

$$\begin{cases} X = \oslash \\ Y = \Re \times \{out_1\} \\ S = \Re \times \Re_0^+ \\ \lambda(\eta, \sigma) = (\eta, out_1) \\ ta(\eta, \sigma) = \sigma \end{cases}$$

La transición interna es:

$$\delta_{int}((\eta, \sigma), r) = (\tilde{\eta}, h)$$

con

$$\begin{cases} \tilde{\eta} = [\sqrt{-2\log(r_1)}\cos(2\pi r_2)]\sqrt{v_\eta} + \mu_\eta \\ r \in \Re^2, r \sim U(0,1)^2, r = (r1, r2) \\ h = \text{sampling interval} \\ \mu_\eta = \text{valor medio esperado del ruido} \\ v_\eta = \text{varianza del ruido} \end{cases}$$

La función de transición externa será $\delta_{ext}(s, e, x, r) = \oslash$ independiente de r.

En este caso, el cálculo de valores sucesivos de $\eta(t)$ se resuelve en δ_{int} usando la transformada de Box-Müller [BM58], un procedimiento bien conocido y preciso [14] para generar números aleatorios independientes según una distribución Normal partiendo de una fuente de números aleatorios distribuidos uniformemente.

Network Induced Random Delay: Se proveerá la definición DEVS–RND medible para M_D^{NIRD}:

$$M_D^{NIRD} = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta)$$

El estado tiene la forma $s = (y', \sigma)$, donde $y' \in \Re$ representa la versión muestreada de la señal de salida con Ruido. Luego,

[14] La transformada de Box-Müller es sólo una de varias aproximaciones posibles [PTVF07, Knu97], con sus ventajas y desventajas a la hora de implementarse algorítmicamente. Dicho análisis queda fuera del alcance de la presente Tesis.

$$\begin{cases} X = \Re \times \{inp_1\} \\ Y = \Re \times \{out_1\} \\ S = \Re \times \Re_0^+ \\ \lambda(y', \sigma) = (y', out_1) \\ ta(y, \sigma) = \sigma \end{cases}$$

La transición interna será:

$$\delta_{int}((y', \sigma), r) = (y', \infty)$$

independiente de r, y la transición externa será:

$$\delta_{ext}((y', \sigma), e, (x, inp_1), r) = (x, \tilde{\sigma})$$

con

$$\begin{cases} \tilde{\sigma} = r \cdot \tau_{max} \\ r \in \Re, r \sim U(0, 1) \\ \tau_{max} = \text{cota superior del retardo de red} \end{cases}$$

Esta definición inherentemente implica interrupciones (o "preemption"). Si una transición externa se dispara a causa de la llegada de la muestra $k+1$–ésima *antes* de que el tiempo de vida $ta(y_k, \sigma_k)$ de la muestra k–ésima haya finalizado, el estado conmutará inmediatamente desde s_k hacia $s_{k+1} = \delta_{ext}(\cdot) = (x_{k+1}, \tilde{\sigma_{k+1}})$, interrumpiendo así el procesamiento de la muestra k. Sin embargo, como se dijo antes, debido a la condición $\tau_{max} < h$ impuesta para $\tau_{net}(t) \forall t$ no existe ninguna chance de que la muestra $k+1$–ésima arribe a este componente antes de que la acción de delay sea completada para la muestra k–ésima.

Simulaciones y Resultados

En esta sección se mostrará una comparación entre los resultados de simulación obtenidos utilizando Matlab y PowerDEVS. Para ambos casos, se barrió el parámetro de simulación τ_{max} desde 0.1 sec. a 0.8 sec. con incrementos de 0.1 segundos. Luego, dado que el período de muestreo es h=1 seg., el peor caso considerado para el retardo inducido por la red the será del 80 % del período del reloj digital del sistema. Para $0{,}8h < \tau_{max} < h$ se verificó una excesiva varianza de la función de costo, mientras que el sistema tiende a inestabilizarse, con lo cual estos casos fueron excluidos ya que no ofrecen ninguna información interesante para los objetivos de comparación entre modelos y herramientas.

Implementación en Matlab

Para el modelo en Matlab–Simulink se usaron bloques de construcción estándar provistos por las bibliotecas *Continuous*, *Discrete* y *Sources*. Se describirán aquí algunas propiedades para los componentes de mayor interés (para más detalles consultar in [CK08b]).

La planta o *Continuous Time Plant* $G_p = \frac{1}{s^2+0{,}8s}$ se describe usando un bloque *Transfer Function*. El *Controller* y *Actuator* de este sistema se describe simplemente por el bloque de sustracción que calcula la señal de error del sistema, es decir, la diferencia entre la señal *Reference* y la versión muestreada y retrasada de la salida ruidosa de la Planta.

Para el componente *AWGN Generator* se usó un bloque *Band–Limited White Noise*, el cual produce una secuencia aleatoria de números con un tiempo de correlación t_c seleccionable, que puede simular el efecto del ruido blanco si t_c se hace mucho más pequeño que la menor constante de tiempo (es decir, la inversa de la parte real del autovalor más veloz) del sistema. Siguiendo la regla práctica sugerida para simulaciones precisas, se eligió $t_c = 0{,}1$ de modo que se cumpla $t_c < \frac{2\pi}{100 f_{max}}$, con f_{max} expresada en rad/seg. Mientras que la covarianza del verdadero ruido blanco es infinita, la aproximación usada en este bloque produce una covarianza (cov) que coincide con su potencia de ruido, que es seleccionable con el parámetro de bloque *NoisePower* dividido por t_c (esto es una conversión de un espectro continuo de densidad de potencia a una covarianza de ruido discreto). Acorde a esto, se elige $NoisePower = cov * t_c$, lo que en el caso de ruido AWGN será $NoisePower = v_\eta * t_c = 0{,}001 t_c = 0{,}0001$ dado que $v_\eta = cov_\eta$ para una distribución normal de esperanza cero.

Para el generador *Network Random Delay* se utiliza un bloque *Uniform Random Number* que reproducirá una tira de números distribuidos uniformemente entre 0 y el parámetro de cota superior de retardo de red τ_{max}.

El ciclo de reloj elegido para la parte discreta del modelo viene impuesta por el bloque *Zero-Order Hold* que cumple las funciones de los componentes *Sample* y the *Hold* del modelo conceptual en la Figura 3.6.

Función de Costo

Las funciones de costo se definen usualmente para dar una medida de la *Calidad de Desempeño* (o "Quality of Performance", QoP) del control. Algunas funciones de uso frecuente como *Integral Time Absolute Error* ITAE y *Integral Absolute Error* IAE se calculan basadas en algún tratamiento de la señal de error junto con el tiempo de simulación [FPEN94].

Se utilizará aquí la función de costo *Mean Squared Error* MSE, para comprender y comparar los resultados de simulación para diferentes condiciones de delay de red. La formulación matemática es como sigue:

$$MSE = \frac{1}{T} \int_{t_0}^{t_f} e(t)^2 dt$$

donde t_0 y t_f son los tiempos inicial y final de simulación, respectivamente (con $T = t_f - t_0$), y $e(t)$ es el error entre la referencia y la versión muestreada y retardada de la trayectoria de salida de la planta (ver Figura 3.7).

Análisis de Resultados

Los resultados de simulación se muestran en la Figura 3.8, en donde las simulaciones se ejecutaron con t_f =10000 sec. y cada punto en la figura representa el valor promedio MSE y las barras de dispersion para 15 corridas. Se verifica que las 2 curvas coinciden ajustadamente, validando así la confiabilidad y utilidad de aplicar STDEVS al caso de un modelo híbrido de tipo NCS.

Aún así, existe una ventaja muy importante en la metodología STDEVS por sobre la simulación en tiempo discreto basada en Matlab, en términos de la robustez y del esfuerzo computacional de los resultados.

Con respecto a la robustez, cuando se utiliza Matlab, hubo que ajustar la tolerancia del método de integración numérica (ode45) a 1×10^{-6}. En cual-

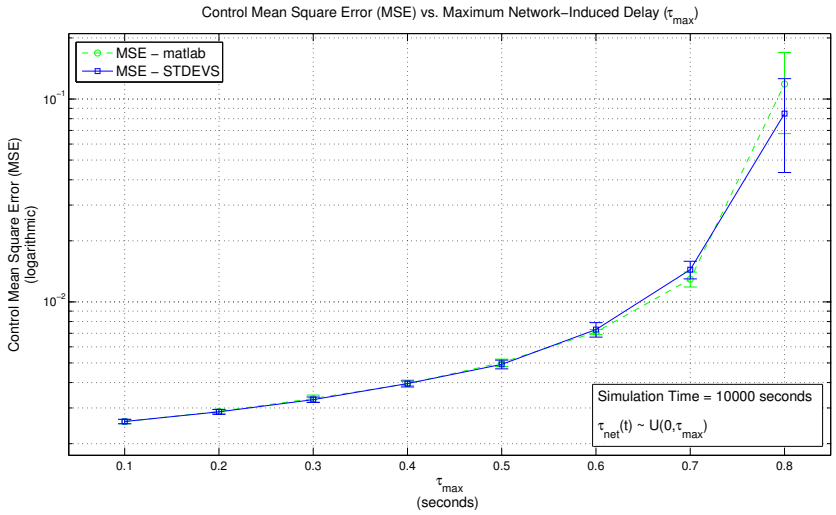


Figura 3.8: Modelo híbrido tipo NCS: Resultados de simulación para Matlab vs. PowerDEVS.

quier otro caso los resultados presentaron errores inaceptables. Esto se debe a la naturaleza híbrida del problema, que combina dinámicas de eventos discretos, tiempo discreto, y tiempo continuo. Cuando se utilizó STDEVS, en cambio, no se requirió hacer ningún ajuste de los parámetros de simulación para producir resultados confiables. Luego, el tratamiento eficiente de eventos se torna crucial.

Con respecto al esfuerzo computacional, cuando se usó PowerDEVS (en donde toda la simulación se basa en el formalismo DEVS), se registraron tiempos de simulación aproximadamente 5.7 veces más veloces que Matlab. Este resultado se obtuvo ejecutando y midiendo ambos simuladores en una plataforma de referencia bajo condiciones controladas.

En todos los casos, para un tiempo virtual de simulación de $T = 50000$ seg. se obtuvo una aceleración de tiempos similar, con lapsos de simulación de tiempo real de aproximadamente $T_{Matlab} = 21{,}5$ seg. para Matlab y $T_{PowerDEVS} = 3{,}75$ seg. para PowerDEVS. La aceleración de la simulación basada en DEVS surge esencialmente porque los métodos numéricos que aproximan la parte continua del sistema (se utilizó el método QSS3) son particularmente eficientes en el manejo de discontinuidades y en la integración numérica de sistemas híbridos [Kof04].

En consecuencia, las simulaciones basadas en DEVS resultaron más rápidas y más confiables que aquellas basadas en la discretización del tiempo.

3.8. Conclusiones

En este capítulo se presentó un nuevo formalismo para describir sistemas generalizados de eventos discretos estocásticos.

La aplicación de la nueva herramienta STDEVS a un caso práctico para el control de congestión de red (modelado discreto del sistema) será realizada en la Sección 7.3.2. Sin embargo, como se hizo evidente en el presente capítulo,

STDEVS trasciende el dominio de aplicación de las redes de datos, constituyendo un avance en el estado del arte de la disciplina de modelado y simulación en general.

Basado en el enfoque de Teoría de Sistemas de DEVS y haciendo uso de la teoría de Espacios de Probabilidad, STDEVS provee un marco formal para modelado y simulación para sistemas de eventos discretos genéricos no deterministas.

STDEVS consiste entonces en un formalismo universal de modelado que provee la capacidad de representar comportamiento estocástico sobre espacios medibles de dimensión infinita, junto con la capacidad de representar formalismos estocásticos tradicionales vastamente utilizados en diversas aplicaciones prácticas (por ejemplo, Cadenas de Markov, Redes de Colas, Redes de Petri Estocásticas). Debido a su raíz basada en DEVS, STDEVS es también una representación basada en Teoría de Sistemas, proveyendo ventajas específicas de modelado interdisciplinario para especificar sistemas híbridos.

Capítulo 4

DEVS y Ecuaciones Diferenciales con Retardos

Continuando con el objetivo de ofrecer soluciones a las limitaciones identificadas en la Sección 2.5.2, en el presente capítulo se presenta una nueva clase de algoritmos de integración para resolver numéricamente Ecuaciones Diferenciales con Retardo (DDE, por *Delay Differential Equations*).

Como se vio antes, en el contexto de las redes de datos, este tipo de sistemas surge al modelar aproximaciones fluidas de controles distribuidos como TCP/AQM en donde la dinámica de las variables en el presente depende de valores de estados en el pasado. Los métodos de integración QSS basados en cuantificación de estados permiten aproximar sistemas continuos bajo el formalismo DEVS, lo cual respeta la línea integradora de formalismos que motiva la presente Tesis. Pero QSS no tiene la capacidad de manipular información histórica de las variables continuas, tal como lo requieren las DDE.

En el presente capítulo se introducen los Sistemas de Estados Cuantificados con Retardo (DQSS, por *Delay Quantized State Systems*), una nueva familia de algoritmos de integración de propósito genérico para resolver numéricamente ecuaciones diferenciales con retardos, basados en técnicas de cuantificación de estados en lugar de la técnica clásica de discretización del tiempo. Se discutirán las propiedades numéricas de estos nuevos algoritmos, es decir, estabilidad y convergencia, y se estudiarán tests comparativos de desempeño de simulaciones contra soluciones del estado del arte reportadas previamente en la literatura abierta.

4.1. Introducción

La solución numérica de modelos de sistemas dinámicos ha sido de gran interés durante décadas. Muchos sistemas físicos pueden ser aproximados por conjuntos de Ecuaciones Diferenciales Ordinarias (ODE), y consecuentemente, la simulación digital de dichos modelos ha captado el interés de ingenieros y matemáticos aplicados desde la invención de la computadora digital. Estos esfuerzos se encuentran resumidos en varios libros de texto [But87, CK06, HNW00, HW91, Lam91].

Sin embargo, existe un número significativo de sistemas en ciencias e ingenierías que requieren, en sus modelos, la inclusión de retardos [BPW95]. A pesar de su importancia, la simulación numérica de modelos descritos por conjuntos Ecuaciones Diferenciales con Retardo (DDE) se encuentra cubierta por muy pocas publicaciones, y también, el estado del arte del software correspondiente está mucho menos desarrollado que el caso de ODE.

Se han desarrollado recientemente buenos algoritmos de integración para DDE (Shampine, Thompson, y colaboradores) llamados dde23 [ST00, ST01] implementados en Matlab. Los autores también proveen un algoritmo en Fortran, llamado dde_solver [S.P08]. Ambos consisten en algoritmos clásicos de integración numérica en el sentido de que están basados en la idea clásica de la discretización del tiempo, utilizada en la mayoría de la literatura para ODE, DDE y Ecuaciones Diferenciales Algebráicas (DAE, por *Algebraic Differential Equations*).

En este capítulo, se busca avanzar el estado del arte de la integración numérica de DDE presentando una nueva clase de algoritmos, denominados DQSS, basados en la cuantificación de las variables de estado en lugar de la discretización del tiempo.

Los algoritmos DQSS fueron implementados en la herramienta de modelado y simulación PowerDEVS [BK10], por medio de la cual se simplifica notablemente el uso de estos algoritmos de integración.

Los algoritmos de integración numérica basados en la cuantificación de estados QSS para sistemas no–rígidos han sido descritos previamente en [CK06, KJ01]. Actualmente, se encuentran disponibles algoritmos para ODE de órdenes 1..4 (QSS1, QSS2, QSS3, y QSS4).

Existe un algoritmo de integración para ODE de sistemas rígidos, con precisión de primer orden, llamado BQSS (backward QSS), descrito en [CKMB08] (cuya discusión formal más profunda acerca de estabilidad y convergencia se encuentra actualmente en preparación). También existe una clase de algoritmos de órdenes superiores, linealmente implícitos, para resolver sistemas rígidos (reportados en [Mig10, MK09]) consistiendo en LIQSS1, LIQSS2 y LIQSS3 para órdenes 1..3 respectivamente. Asimismo, existe un algoritmo de integración de primer orden para resolver ODE marginalmente estables, denominados CQSS (centered QSS), descritos previamente en [CKMB08]. El método CQSS es simétrico y simpléctico y preserva reversibilidad–ρ [HLW02]. En [Kof04] puede encontrarse una discusión acerca del manejo de discontinuidades (solución de raíces) en los algoritmos de la familia QSS.

Los métodos QSS resultan ser muy útiles para la simulación de modelos DDE. Existe un número de razones para esta observación. QSS ofrece salida densa, preservando dicha información no solo durante un paso de integración, sino durante la simulación completa. Las trayectorias de QSS consisten en secuencias de segmentos polinómicos. Los coeficientes de estos segmentos polinómicos cambian únicamente en instantes de eventos, es decir, cuando una variable de estado cambia su nivel de cuantificación. Para reconstruir la trayectoria de una solución, es suficiente ir almacenando cada instante inmediato futuro en el cual los coeficientes polinómicos cambiarán, permaneciendo válida, hasta tanto, la descripción polinómica actual de la solución. Procediendo de este modo, la solución completa consistirá en una secuencia finita de secciones polinómicas, de duración finita. La información para cada variable de estado puede ser mantenida por separado, dado que los métodos QSS son intrínsecamente asíncronos

(es decir, cada variable de estado se actualiza independientemente del resto del sistema, cada vez que la misma cruza el siguiente umbral de cuantificación). Estas características pueden ser aprovechadas para el cálculo (interpolación) de señales con retardos.

Este capítulo se organiza del siguiente modo. Luego de una breve introducción a la cuantificación de estados, en la Sección 2.3.1 se presentará la idea básica de los métodos QSS. En la Sección 4.2 se discutirá la generalización del concepto de QSS aplicado a DDE y se explicarán los algoritmos DQSS en detalle. La Sección 4.3, junto con los Apéndices A.1 y A.3, discutirán las propiedades numéricas de los algoritmos DQSS, es decir, estabilidad y convergencia. Finalmente la Sección 4.4 presenta simulaciones DQSS en cuatro estudios comparativos de desempeño, aplicados a modelos tomados de la literatura abierta [ST01, S.P08].

4.1.1. Idea Intuitiva

Se seguirá un procedimiento paso a paso basado en un enfoque de cuantificación de estados para resolver un ejemplo típico de DDE tomado de [BTE09]. Se pretende simular el siguiente sistema:

$$\begin{aligned} \dot{x}(t) &= x(t-1)\ ,\ t \geq 0 \\ x(t) &= 1\ ,\ -1 < t \leq 0 \end{aligned} \tag{4.1}$$

para $t \geq 0$, con historia inicial $x(t) = 1$. Se trata de una DDE de primer orden con un delay constante $\tau = 1$.

Se considerará la siguiente función de cuantificación:

$$Q(x) \triangleq floor[\frac{x}{\Delta Q}].\Delta Q \tag{4.2}$$

donde ΔQ es un parámetro denominado *quantum*.

En la Figura 4.1, se muestra la relación de Ec. (4.2) para un quantum $\Delta Q =$ 0,5.

Ahora se aproximará al sistema original de Ec. (4.1) como sigue:

$$\begin{aligned} \dot{x}(t) &= Q(x(t-1)) \triangleq q(t-1)\ ,\ t \geq 0 \\ x(t) &= 1\ ,\ -1 < t \leq 0 \end{aligned} \tag{4.3}$$

es decir, se reemplaza $x(\cdot)$ por $q(\cdot) \triangleq Q(x(\cdot))$ en la parte a la derecha de la ecuación de estados. La variable $q(t)$ se denomina *quantized variable*.

Notar que $Q(1) = 1$ por lo cual $q(t) = 1$ para $t \leq 0$. Luego, puede reescribirse Ec. (4.3) como sigue:

$$\begin{aligned} \dot{x}(t) &= q(t-1)\ ,\ t \geq 0 \\ q(t) &= 1\ ,\ -1 < t \leq 0 \end{aligned} \tag{4.4}$$

El sistema modificado de Ec. (4.4) puede ser resuelto analíticamente.

Se seguirá ahora un procedimiento intuitivo de resolución tomando como guía la Figura 4.2, donde se muestra la trayectoria solución $x(t)$ (línea punteada), la

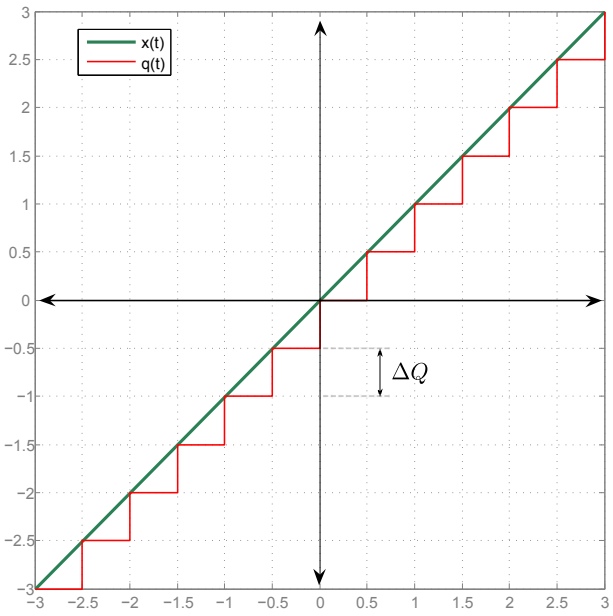


Figura 4.1: Función de cuantificación $q(x)$ con quantum $\Delta Q = 0{,}5$

variable cuantizada $q(t)$ (línea llena), y su versión retardada $q(t-1)$ (línea de punto–raya).

Comenzando en $t = t_0 = 0$ con $x(0) = q(0) = 1$. En este punto, se tiene $\dot{x}(t=0) = q(-1) = 1$.

Puede observarse que $\dot{x}(t)$ no cambia hasta que $q(t-1)$ cambia. Luego, $x(t)$ crece transitoriamente con una pendiente constante igual a 1. Luego, en el instante $t = t_1 = 0{,}5$, se tiene $x(t = 0{,}5) = 1{,}5$ y la variable cuantizada $q(t)$ experimenta su primer cambio, tomando el nuevo valor $q(t = 0{,}5) = 1{,}5$. Esto implica que $\dot{x}(t) = q(t-1)$ permanecerá sin cambios hasta el instante $t = 1{,}5$.

En $t = t_2 = 1$, se tiene $x(t = 1) = 2$ y consecuentemente, $q(t)$ cambia su valor nuevamente a $q(t = 1) = 2$. Luego, en $t = t_3 = 1{,}5$, el estado alcanza un valor de $x(t = 1{,}5) = 2{,}5$, y el estado cuantizado cambia su valor a $q(t = 1{,}5) = 2{,}5$.

En el instante $t = 1{,}5$, la derivada $\dot{x}(t)$ del estado cambia también su valor (dado que $q(t)$ cambió en $t = 1{,}5 - 1 = 0{,}5$). La nueva pendiente es $\dot{x}(t > 1{,}5) = q(0{,}5) = 1{,}5$, y ahora $x(t)$ crece a mayor velocidad.

Luego de $\frac{\Delta Q}{\dot{x}(t_3)} = \frac{0{,}5}{1{,}5}$ unidades de tiempo, es decir, en el instante $t_4 = t_3 + \frac{1}{3} = 1{,}8333$, el estado alcanza un valor de $x(t_4) = 3$, y se tendrá $q(t_4) = 3$.

Nuevamente, la derivada del estado $\dot{x}(t)$ cambia en $t = t_5 = 2$, y desde ese instante en adelante, se tiene $\dot{x}(t) = 2$, dado que $q(2-1=1) = 2$. En el instante t_5, el estado alcanza el valor $x(t_5) = x(t_4) + 1{,}5 \cdot (t_5 - t_4) = 3{,}25$.

Desde la nueva condición $x(t) = 3{,}25$ y $\dot{x}(t) = 2$, el estado evoluciona y alcanza el valor $x(t = 2{,}125) = 3{,}5$ luego de $\frac{3{,}5-3{,}25}{2} = 0{,}125$ unidades de tiempo, es decir, en el instante $t_6 = 2 + 0{,}125 = 2{,}125$. Luego, $q(t)$ cambia su valor nuevamente, y así los cálculos pueden continuarse con la misma mecánica hasta el final de la simulación.

Pueden distinguirse dos tipos de cambio distintos. En ciertos instantes: t_1, t_2, t_3, t_4, y t_6, el estado cuantizado $q(t)$ cambia su valor. En cambio, en otros

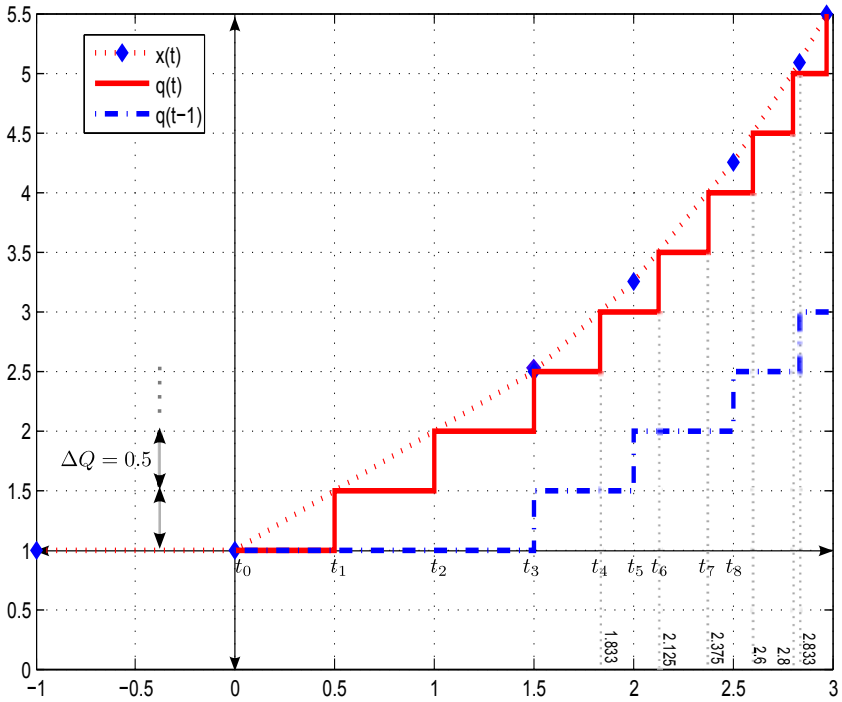


Figura 4.2: Solución de $x(t)$ para una DDE aproximada por la Ec. (4.4).

instantes: t_3 y t_5, es la derivada $\dot{x}(t)$ la que cambia su valor. Incluso en t_3, ambos tipos de cambios ocurren simultáneamente. El comportamiento observado sugiere que el sistema aproximado por la Ec. (4.4) puede ser descrito usando un modelo a eventos discretos.

A pesar de que el procedimiento para simular al sistema aproximado de Ec. (4.4) parece ser bastante directo, queda pendiente establecer qué tan bien puede aproximarse la solución del sistema original Ec. (4.1) por medio de la solución del sistema aproximado. Aún más, a pesar de que pareció ser bastante sencillo simular este sistema particular, se debe discutir si es posible (y si lo es, de qué manera) el enfoque propuesto puede ser generalizado para atacar clases generales de sistemas DDE.

Los métodos de integración numérica por cuantificación de estados QSS presentados en 2.3 formalizan y generalizan el procedimiento intuitivo que se presentó en esta sección. En la siguiente Sección 4.2, se estudiarán los requisitos que surgen cuando se intenta extender la aplicación de QSS desde su contexto original (la aproximación de ODE) al nuevo contexto de aproximación de DDE. Más adelante se analizarán rigurosamente las limitaciones teóricas que presenta la aplicación de QSS para DDE.

4.2. Métodos QSS para Ecuaciones Diferenciales con Retardo

En esta sección, se presentarán los métodos Delay Quantized State System (DQSS) para resolver numéricamente DDE.

Las DDE se caracterizan por su capacidad de hacer referencia a valores pasados de las variables de estado. Las DDE expresan la información del pasado (la historia de los estados) por medio de una o más *funciones retardo* (o "delay functions") de la forma genérica $\tau_j(\cdot)$. Los retardos pueden ser tanto constantes del tipo $\tau_j(\cdot) = \tau_j$ (constant–delay DDE), o pueden ser funciones del tiempo de tipo $\tau_j(\cdot) = \tau_j(t)$ (time–dependent DDE). Incluso los retardos pueden depender los propios estados del sistema, acorde a $\tau_j(\cdot) = \tau_j(x_1(t), ..., x_n(t), t)$ (state–dependent DDE).

Se reformulará ahora la representación genérica de ODE en Ec. (2.1) para incorporar el tipo de información retardada mencionada más arriba, y luego se aplicará una función de cuantificación a las variables de estado, para obtener así lo que se llamará *métodos DQSS*.

4.2.1. Definición de DQSS

Considérese la siguiente DDE en su representación de espacio vectorial de estados (o State Equation System - SES):

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{x}(t - \tau_1(\mathbf{x}, t)), ..., \mathbf{x}(t - \tau_m(\mathbf{x}, t)), \mathbf{u}(t)) \tag{4.5}$$

donde $\tau_j(\mathbf{x}, t) \geq 0$ para todo $t, \mathbf{x}$ y para todo j, y el resto de las variables y funciones quedan definidas como en el caso de ODE en la Ec. (2.1).

Considérese también la *historia inicial*:

$$\mathbf{x}(t) = \mathbf{S}(t) \quad ; \quad -\tau_{max} < t < 0 \tag{4.6}$$

donde τ_{max} es una constante positiva que satisface $t - \tau_i(\mathbf{x}, t) > -\tau_{max}$ para todo $t, \mathbf{x}$ y para todo i.

Un método DQSSn simula el sistema *retardado y cuantificado*:

$$\dot{\mathbf{x}}(t) = f(\mathbf{q}(t), \mathbf{q}(t - \tau_1(\mathbf{q}, t)), ..., \mathbf{q}(t - \tau_m(\mathbf{q}, t)), \mathbf{u}(t)) \tag{4.7}$$

donde cada componente del vector de estados cuantificados $\mathbf{q}(\mathbf{t})$ está relacionado uno a uno con su componente correspondiente en el vector de estados $\mathbf{x}(\mathbf{t})$ por medio de una función de cuantificación con histéresis de orden n.

4.2.2. Representación DEVS de DQSS

La Figura 4.3 muestra el diagrama de bloques del sistema retardado y cuantificado de la Ec. (4.7).

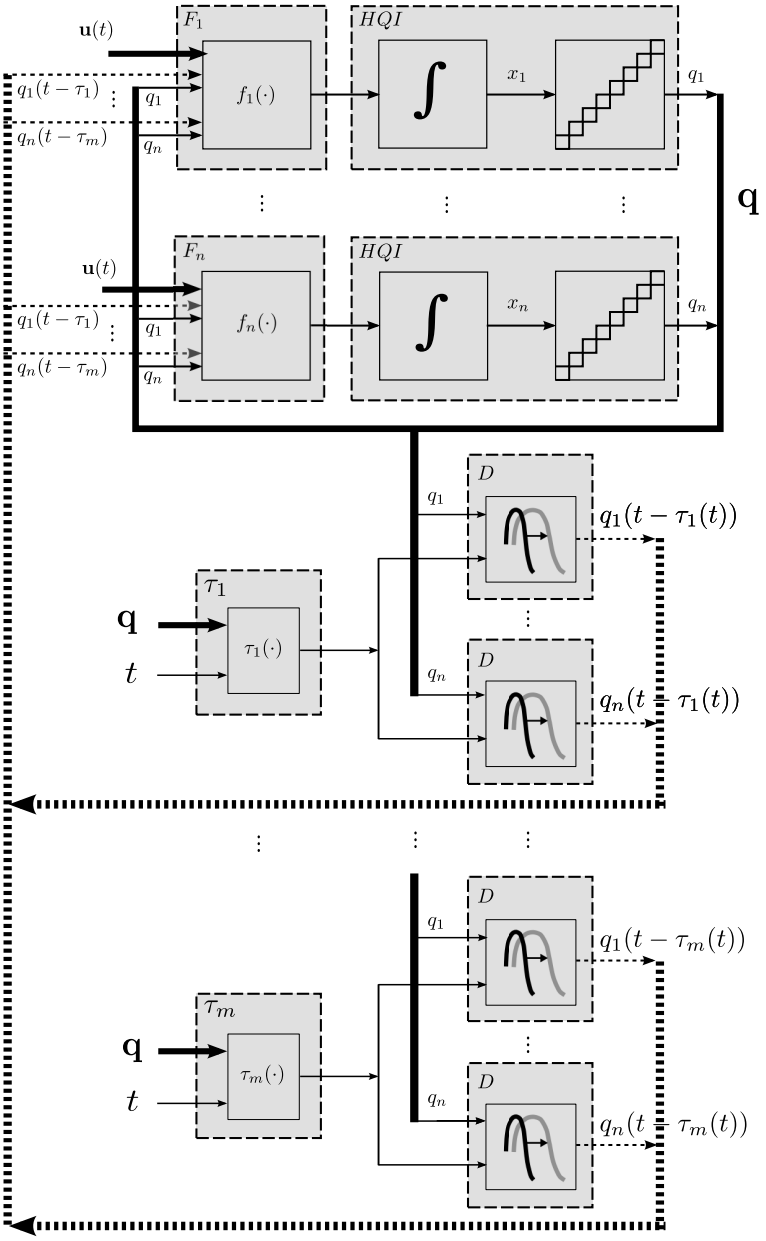


Figura 4.3: Representación en diagrama de bloques de un sistema DQSS. Referencias de líneas de cajas: continua (bloque estándar), punteada (modelo DEVS equivalente). Referencias de flechas: fina (señal simple), gruesa (bus de múltiples señales), continua (señal no retardada), punteada (señal retardada).

Para implementar este método utilizando DEVS, puede procederse siguiendo la misma idea utilizada para el caso de ODE, es decir, se construirá un modelo acoplado DEVS con los equivalentes atómicos DEVS de los bloques del sistema.

El diagrama de bloques de la Figura 4.3 muestra ahora tres tipos de modelos DEVS: *Hysteretic Quantized Integrators HQI*, que calculan q_j; *Static Functions* F_j y τ_i, que calculan $\dot{x}_j$ y τ_i respectivamente, y *Delays* D, que calculan $\mathbf{q}(t-\tau_i)$.

Dado que ya se dipone de modelos DEVS para integradores cuantificados u para funciones estáticas, el único nuevo requisito es encontrar una representación DEVS equivalente para los bloques de retardo D. El primer paso hacia este objetivo es definir matemáticamente la operatoria de un bloque genérico de retardo (Figura 4.4, izquierda). Luego de ello, se deberá encontrar el modelo DEVS equivalente D (Figura 4.4, derecha) que implemente la definición matemática obtenida.

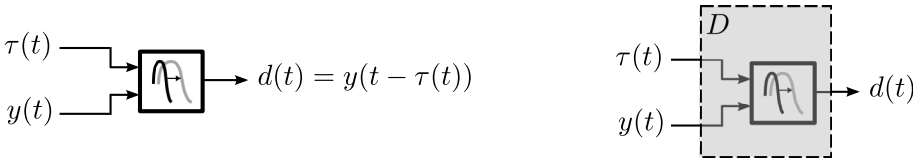


Figura 4.4: Bloque Delay (izquierda). Modelo DEVS Delay D (derecha)

4.2.3. Cálculo Analítico de una Señal Retardada

Es necesario en primer lugar hallar una expresión analítica para $d(t) = y(t-\tau(t))$, donde $y(t)$ y $\tau(t)$ son trayectorias polinomiales a tramos.

Es posible expresar $y(t)$ como una secuencia de segmentos polinomiales, válidos en sucesivos intervalos adyacentes de tiempo, acorde a:

$$\begin{aligned} &y(t) = y_{0,k} + y_{1,k}(t-t_k^y) + y_{2,k}(t-t_k^y)^2 + \dots \\ &t_k^y \le t < t_{k+1}^y \end{aligned} \tag{4.8}$$

donde t_k^y es una secuencia de instantes con $t_k^y < t_{k+1}^y$, y $y_{0,k}, y_{1,k}, y_{2,k}, \dots$ son las secuencias correspondientes de coeficientes polinomiales.

Notar que con esta representación propuesta, los coeficientes pueden cambiar sus valores únicamente al comienzo de cada intervalo (t_k^y), y permanecen constantes durante cada lapso $t \in [t_k^y, t_{k+1}^y)$.

De modo similar, se expresará $\tau(t)$ como la siguiente secuencia:

$$\begin{aligned} &\tau(t) = \tau_{0,j} + \tau_{1,j}(t-t_j^\tau) + \tau_{2,j}(t-t_j^\tau)^2 + \dots \\ &t_j^\tau \le t < t_{j+1}^\tau \end{aligned} \tag{4.9}$$

donde t_j^τ es una secuencia de instantes con $t_j^\tau < t_{j+1}^\tau$, y $\tau_{0,j}, \tau_{1,j}, \tau_{2,j}, \dots$ son las correspondientes secuencias de coeficientes polinomiales.

La señal retardada $d(t)$ también seguirá una trayectoria polinomial que puede escribirse como:

$$\begin{aligned} &d(t) = d_{0,\ell} + d_{1,\ell}(t-t_\ell^d) + d_{2,\ell}(t-t_\ell^d)^2 + \dots \\ &t_\ell^d \le t < t_{\ell+1}^d \end{aligned} \tag{4.10}$$

siendo entonces necesario calcular las secuencias temporales t_ℓ^d y las secuencias de coeficientes $d_{0,\ell}$, $d_{1,\ell}$, $d_{2,\ell}$, ...

Notar que no existe una relación preestablecida entre los instantes t_k^y y t_j^τ, pudiendo evolucionar independientemente. La secuencia temporal resultante t_ℓ^d será función de t_k^y y de t_j^τ de una manera que se describirá a continuación.

Sea t^* un instante arbitrario de tiempo, para el cual se desea encontrar la expresión de $d(t^*)$. Sea j y k números enteros que satisfacen:

$$t_j^\tau \leq t^* < t_{j+1}^\tau$$
$$t_k^y \leq t^* - \tau(t^*) < t_{k+1}^y$$

Cabe notar que mientras una DDE utiliza valores pasados de sus variables de estado para definir su evolución actual, la función de retardo $\tau(t)$ se evalúa sólo en el "presente", es decir, en aquellos instantes en que se "accede" a la información del pasado, y no cuando dicha información fue calculada.

También nótese que $d(t^*)$ será relacionado tanto con la j–ésima sección polinomial de $\tau(t)$ como con la k–ésima sección polinomial de $y(t)$. El comienzo de la ℓ–ésima sección polinomial de $d(t)$ puede corresponder tanto al evento de un cambio de sección en $\tau(t)$, o a un cambio de sección en $y(t)$, o a ambos simultáneamente. Luego, t_ℓ^d estará relacionado siempre con alguno de los instantes t_j^τ, t_k^y o t_{k+1}^y.

Defínase ahora:

$$\begin{aligned} t_a &= \text{máx}\left(t \mid (t - \tau(t) = t_k^y) \wedge (t_j^\tau < t \leq t^*)\right) \\ t_b &= \text{máx}\left(t \mid (t - \tau(t) = t_{k+1}^y) \wedge (t_j^\tau < t \leq t^*)\right) \end{aligned} \tag{4.11}$$

en el caso de que las ecuaciones $t - \tau(t) = t_k^y$ y/o $t - \tau(t) = t_{k+1}^y$ tengan solución en el intervalo dado. De otro modo, se considerará $t_a = -\infty$ y/o $t_b = -\infty$.

Luego, están dadas las consiciones para calcular el tiempo de inicio $t_\ell^d < t^*$ de la sección polinomial ℓ–ésima a la cual pertenece $d(t^*)$ como:

$$t_\ell^d = \text{máx}(t_a, t_b, t_j^\tau) \tag{4.12}$$

La siguiente sección adyacente a $d(t)$ comenzará en $t_{\ell+1}^d$. Para calcular su valor, se definirá:

$$\begin{aligned} t'_a &= \text{mín}\left(t \mid (t - \tau(t) = t_k^y) \wedge (t^* < t \leq t_{j+1}^\tau)\right) \\ t'_b &= \text{mín}\left(t \mid (t - \tau(t) = t_{k+1}^y) \wedge (t^* < t \leq t_{j+1}^\tau)\right) \end{aligned} \tag{4.13}$$

en caso de que las ecuaciones $t - \tau(t) = t_k^y$ y/o $t - \tau(t) = t_{k+1}^y$ tengan solución en el intervalo dado. De otro modo, se considerará $t'_a = \infty$ y/o $t'_b = \infty$.

El instante subsiguiente $t_{\ell+1}^d > t^*$ que pertenece a la secuencia de la señal retardada $d(t)$ puede calcularse como:

$$t_{\ell+1}^d = \text{mín}(t'_a, t'_b, t_{j+1}^\tau) \tag{4.14}$$

Siguiendo estas definiciones, y para cada instante perteneciente al intervalo continuo $t \in [t^d_\ell, t^d_{\ell+1})$, no existirán discontinuidades en las trayectorias $y(t-\tau(t))$ y $\tau(t)$, y se cumplirá que $d(t)$ satisface la Ec. (4.10).

En la Figura 4.5, se describen los principales instantes y secuencias polinómicas discutidas hasta aquí que definen unívocamente a $d(t) = y(t-\tau(t))$. De aquí en adelante, se hará referencia a los coeficientes polinomiales como vectores columna, y dado que el interés principal será en métodos de tercer orden (QSS3), se manejarán vectores de tres coeficientes (es decir, hasta el término cuadrático inclusive).

Luego, se manipularán los vectores:

$$\mathbf{d}_\ell = \begin{bmatrix} d_{0,\ell} \\ d_{1,\ell} \\ d_{2,\ell} \end{bmatrix}, \quad \mathbf{y_k} = \begin{bmatrix} y_{0,k} \\ y_{1,k} \\ y_{2,k} \end{bmatrix}, \text{ y } \boldsymbol{\tau_j} = \begin{bmatrix} \tau_{0,j} \\ \tau_{1,j} \\ \tau_{2,j} \end{bmatrix}$$

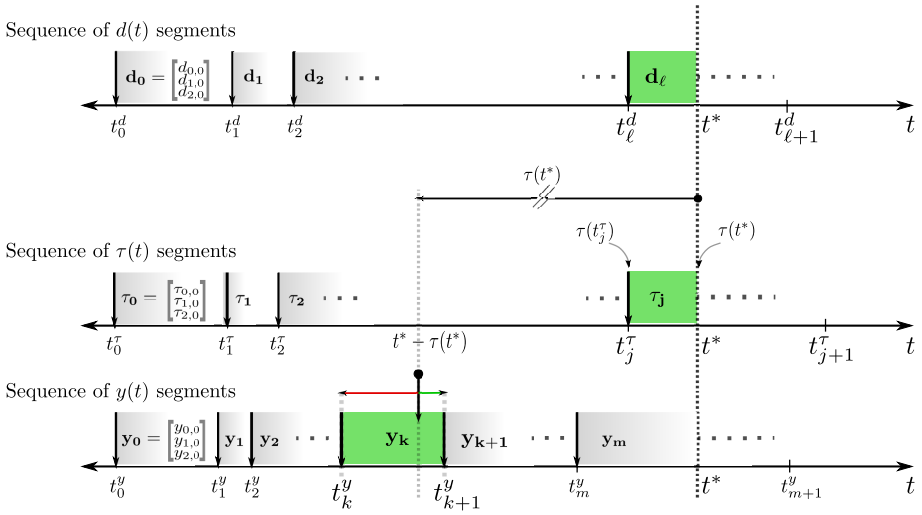


Figura 4.5: Algoritmo DQSS basado en segmentos polinomiales: secuencias de instantes y coeficientes polinomiales relevantes.

Tanto la Ec. (4.12) como la Ec. (4.14) pueden usarse para definir la secuencia temporal t^d_ℓ de la función retardada $d(t)$. Para completar su expresión analítica, es necesario calcular la secuencia de coeficientes $d_{0,0} \dots d_{0,\ell}$, $d_{1,0} \dots d_{1,\ell}$, y $d_{2,0} \dots d_{2,\ell}$. De las Ec. (4.8), (4.9) y (4.10), puede escribirse:

$$\begin{aligned} d(t) &= d_{0,\ell} + d_{1,\ell}(t - t^d_\ell) + d_{2,\ell}(t - t^d_\ell)^2 + \dots = \\ &= y(t - \tau(t)) = \\ &= y_{0,k} + y_{1,k}[(t - \tau(t)) - t^y_k] + y_{2,k}[(t - \tau(t)) - t^y_k]^2 + \dots = \\ &= y_{0,k} + y_{1,k}[(t - (\tau_{0,j} + \tau_{1,j}(t - t^\tau_j) + \tau_{2,j}(t - t^\tau_j)^2 + \dots)) - t^y_k] + \\ &\quad + y_{2,k}[(t - (\tau_{0,j} + \tau_{1,j}(t - t^\tau_j) + \tau_{2,j}(t - t^\tau_j)^2 + \dots)) - t^y_k]^2 + \dots \end{aligned}$$

Quitando todos los coeficientes de órdenes superiores a 2 y eliminando los

subíndices j, k y ℓ (para aportar claridad a la notación), se obtienen las siguientes expresiones para los coeficientes:

$$\begin{aligned}
d_0 =& y_0 - y_1[\tau_0 + \tau_1(t^d - t^\tau) + \tau_2((t^\tau)^2 - 2t^\tau t^d + (t^d)^2) + t^y - t^d] + \\
& y_2[\tau_0^2 - \tau_0(2\tau_1(t^\tau - t^d) - \tau_2(2(t^\tau)^2 - 4t^\tau t^d + 2(t^d)^2) - 2t^y + 2t^d) + \\
& \tau_1^2 t^\tau - (t^d)^2 + 2\tau_1(t^d - t^\tau)(\tau_2((t^\tau)^2 - 2t^\tau t^d + (t^d)^2) + t^y - t^d) + \\
& \tau_2^2((t^\tau)^2 - 2t^\tau t^d + (t^d)^2)^2 + 2\tau_2((t^\tau)^2 - 2t^\tau t^d + (t^d)^2)(t^y - t^d) + \\
& (t^y)^2 - 2t^y t^d + (t^d)^2] \\
d_1 =& - y_1[\tau_1 + 2\tau_2(t^d - t^\tau) - 1] + \\
& y_2 2[\tau_0 + \tau_1(t^d - t^\tau) + \tau_2((t^\tau)^2 - 2t^\tau t^d + (t^d)^2) + t^y - t^d] \cdot \\
& [\tau_1 + 2\tau_2(t^d - t^\tau) - 1] \\
d_2 =& - y_1\tau_2 + y_2[2\tau_0\tau_2 + \tau_1^2 - \tau_1(6\tau_2(t^\tau - t^d) + 2) + 6\tau_2^2((t^\tau)^2 - 2t^\tau t^d + (t^d)^2) + \\
& 2\tau_2(2t^\tau + t^y - 3t^d) + 1]
\end{aligned} \tag{4.15}$$

4.2.4. Algoritmo basado en DEVS para obtener $d(t) = y(t - \tau(t))$

Dada la naturaleza basada en eventos de los sistemas QSS, la tarea de retardar segmentos se reduce a una actividad de encolamiento de información. Los eventos $y(t)$ que arriban (descripciones polinomiales de segmentos) deben ser recibidos, encolados, posiblemente consultados (tantas veces como se requiera, acorde a la evolución de $\tau(t)$), procesados por un algortimo de transformación, y finalmente enviados en forma de nuevos eventos $d(t)$.

En la Sección 4.1.1, se presentó una procedimiento iterativo informal usando un simple criterio de búsqueda hacia atrás para acceder a la información $q(t-\tau)$ partiendo de $q(t)$. Resultó ser bastante sencillo de comprender y no presentó ninguna complejidad algorítmica. Pero el sistema resuelto en dicho ejemplo es uno muy simple, dado que consta de un único retardo, el mismo es constante, y además se utilizó un método de integración de primer orden. Si bien existen muchos modelos de interés práctico que involucran retardos constantes, los modelos con retardos dependientes del tiempo y del estado son muy importantes, siendo los problemas más desafiantes para los métodos clásicos de integración a tiempo discreto para aproximar DDE.

Los modelos DEVS tipo Delay D deben aceptar información de retardo variable, y deben implementar un mecanismo para buscar dinámicamente hacia atrás en el tiempo a través de los segmentos $y(t)$ almacenados acorde a lo indicado por $\tau(t)$, como se discutió en Sección 4.2.3.

Se describirá entonces una implementación genérica del modelo DEVS Delay D que obedece la especificación matemática provista en la Sección 4.2.3. Las principales actividades de D se describirán en concordancia a la descripción de la Figura 4.5, y pueden sintetizarse como sigue:

- **Transiciones externas DEVS**[1]:

[1] Ante la llegada de múltiples nuevos segmentos $y(t)$ y/o $\tau(t)$ *simultáneamente*, se dis-

- Al recibir un nuevo segmento $y(t)$ en el instante t, hacer:

  Encolar el nuevo segmento como un vector $\mathbf{y_m}$ con estampa de tiempo $t_m^y = t$. El mismo contiene n coeficientes polinomiales describiendo el m–ésimo segmento de la trayectoria $y(t)$.

- Al recibir un nuevo segmento $\tau(t)$ en el instante t, hacer:

  Actualizar el vector $\tau_\mathbf{j}$ almacenado antes con el nuevo segmento, con estampa temporal t_j^τ. El mismo contiene los n coeficientes polinomiales que describen al j–ésimo segmento de la trayectoria $\tau(t)$.

  Buscar hacia atrás en la cola el k–ésimo segmento $\mathbf{y_k}$ con estampa temporal t_k^y "apuntada por" $\tau(t)$, es decir, aquella que satisface $t_k^y \leq t - \tau(t) < t_{k+1}^y$.

- Luego de haber procesado todos los eventos externos que arribaron en el tiempo t, hacer:

  Calcular el próximo instante $t_{\ell+1}^d$ acorde a la Ec. (4.14) en el cual expira la validez de la salida actual $\mathbf{d}_\ell$.

  Agendar una transición interna para que ocurra en $t = t_{\ell+1}^d$ para calcular $t_{\ell+2}^d$, asumiendo que no interfiera una transición externa.

- **Transiciones internas DEVS**:

  - Al expirar el segmento actualmente en la salida $\mathbf{d}_\ell$ en el instante $t = t_{\ell+1}^d$, hacer:

    Recalcular y Emitir un nuevo vector $\mathbf{d_{\ell+1}}$ con estampa temporal $t_{\ell+1}^d$ y coeficientes polinómicos según la Ec. (4.15).

    Calcular y Agendar una nueva transición interna para el instante $t = t_{\ell+2}^d$, para entonces calcular $t_{\ell+3}^d$, asumiendo que no interfiera una transición interna.

El pseudo–algoritmo recién descrito puede ser fácilmente implementado en un modelo DEVS atómico. En la sección que sigue, se proveerán los detalles de la implementación del modelo Delay realizada en esta Tesis utilizando la herramienta PowerDEVS.

4.2.5. Implementación en PowerDEVS

Se desarrolló un nuevo bloque llamado *Delay* para la biblioteca *Continuous Library* de PowerDEVS. El mismo consiste en un modelo atómico DEVS que implementa el algoritmo descrito en la Figura 4.5.

El bloque Delay tiene dos puertos de entrada y uno de salida. El puerto superior recibe la información de retardo variable $\tau(t)$, y el puerto inferior recibe la señal $y(t)$ a ser retardada. El puerto de salida emite la señal retardada resultante $d(t) = y(t - \tau(t))$.

En la Figura 4.6 se muestra la herramienta PowerDEVS con un bloque Delay y su ventana de interfaz de usuario para parametrización.

---

pararán las transiciones externas consecutivas necesarias para procesar cada evento en forma independiente, una a la vez. El orden de procesamiento de los eventos será el mismo que su orden de llegada al modelo D, que a su vez, queda unívocamente determinado por las prioridades de los modelos que los generaron. El estado final resultante para el modelo D corresponderá al procesamiento del evento último en llegar. Este mecanismo no afecta la correctitud de la secuencia de segmentos de salida $d(t)$.

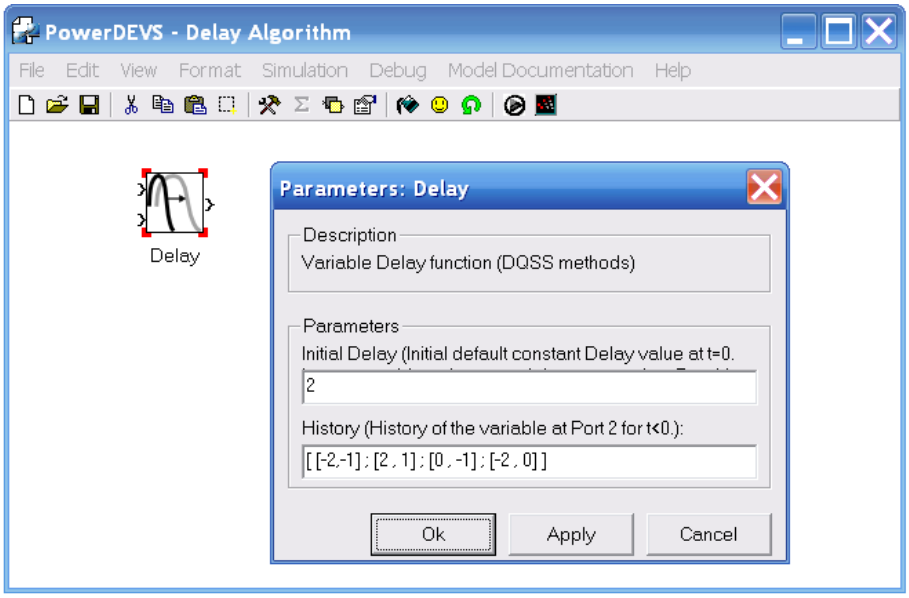


Figura 4.6: Interfaz de usuario del bloque Delay en PowerDEVS.

Se pueden configurar dos parámetros: *Initial Delay* y *History*. Ambos utilizan expresiones o variables Scilab.

Configurando *Initial Delay*, se provee el valor para $\tau_0 = \tau(t_0)$ al comienzo de la simulación, en caso de que no haya una señal presente en el puerto superior en el instante t_0. Por ejemplo, en el ejemplo de la Figura 4.6, el bloque se configura con $\tau_0 = 2$, y en consecuencia comenzará emitiendo $d(t_0) = y(t_0 - \tau_0) = y(-2)$.

Los valores para $y(t \leq 0)$ se ingresan en el parámetro *History*. La información de la historia se ingresa como una matriz $H(r, c)$ de tamaño $r \times c$ con r filas y c columnas. Cada columna define un segmento polinómico y su estampa de tiempo. La primer fila $H(1, \cdot)$ representa la secuencia de estampas de tiempo negativas para cada segmento. Por ejemplo, en el ejemplo de la Figura 4.6, se tiene $H(1, \cdot) = [-2, -1]$, significando que el bloque es provisto con dos segmentos de historia, comenzando en $t = -2$ y $t = -1$ respectivamente.

Para una columna dada $H(\cdot, c)$, la primer fila define la estampa de tiempo, y las filas desde la 2da. hasta la r–ésima representan los coeficientes polinómicos de órdenes $0, 1, 2, \ldots$.

Por ejemplo, en el ejemplo de la Figura 4.6, el segmento más antiguo en $t = -2$ tiene coeficientes $[2, 0, -2]$ y permanecerá válido hasta el comienzo del siguiente segmento, en $t = -1$, cuyos nuevos coeficientes son $[1, -1, 0]$. Con estos parámetros, el bloque comenzará a emitir $d_0 = y(-2) = 2$ en t_0.

Nótese que podría suceder que el parámetro *Initial Delay* y/o $\tau(t)$ apunten a un instante t_{old} en la historia para el cual no exista ningún segmento definido en el parámetro *History*, es decir, cuando $t_{old} < t_{min} = min(h_i | h_i \in H(1, i), i = 1, \ldots, c)$. En esta situación, el bloque Delay extrapola hacia atrás en el tiempo los coeficientes del segmento más antiguo que tiene definido en H, y lo hará de modo que su nueva estampa temporal t'_{min} coincida con t_{old}. Cada vez que esta situación ocurra, se escribirá un mensaje de alerta en el archivo de log de

mensajes.

El bloque Delay puede ser conectado a cualquier bloque PowerDEVS de las bibliotecas Continuous Library y/o Hybrid Library para construir sistemas híbridos y complejos con retardos.

4.3. Propiedades Teóricas de los Métodos QSS para DDEs.

En la presente sección se analizarán las características de estabilidad y convergencia para los casos de DQSS con retardos constantes y luego con retardos variables y dependientes del estado. Las propiedades serán justificadas por medio de teoremas cuyas demostraciones figuran en la sección de apéndices.

4.3.1. Estabilidad y Convergencia. Retardos Constantes.

Con el siguiente teorema se establecen las condiciones que garantizan la aplicabilidad de los métodos DQSS para resolver DDE genéricas de retardo constante.

Se probará que, dado un sistema original estable, la simulación del sistema DQSS equivalente preservará la estabilidad numérica, y también que la solución numérica podrá hacerse tan precisa como se desee haciendo tender a cero el quantum $\mathbf{\Delta Q}$. Solamente se requerirá que el sistema original tenga solución asintóticamente estable para la condición inicial nula.

Teorema 4.1. *Considérese la siguiente DDE:*

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + \sum_{i=1}^{m} A_i \mathbf{x}(t - \tau_i) \tag{4.16}$$

asumiendo que la solución analítica $\phi_{\mathbf{a}}(t)$ para la condición inicial trivial $\phi_{\mathbf{a}}(t) = \mathbf{x}(t) = 0$, $t \leq 0$ es asintóticamente estable.

Luego,

1. *El error cometido por cualquier método QSS en la simulación de la Ec. (4.16) está acotada para todo $t \geq 0$, para cualquier $\mathbf{\Delta Q}$ adoptado, y para cualquier condición inicial $\mathbf{x}(t \leq 0)$.*

2. *La aproximación QSS converge a la solución analítica, es decir, el error global tiende a cero cuando el quantum tiende a cero.*

Demostración. Ver A.2 □

4.3.2. Estabilidad y Convergencia. Retardos Dependientes del Tiempo y del Estado.

Con el próximo teorema, se extiende la clase de DDE que puede ser simulada con métodos DQSS a la clase de sistemas genéricos cuyos retardos son dependientes del tiempo y del estado, garantizando la aplicabilidad de los métodos DQSS.

Como antes, se prueba que la simulación del sistema DQSS equivalente de un sistema original estable preservará la estabilidad numérica, y que al mismo

tiempo la solución numérica puede hacerse tan precisa como se desee haciendo tender a cero el quantum $\mathbf{\Delta Q}$.

Pero en este caso, se requerirá una propiedad más restrictiva sobre el sistema original. Se pedirá que el sistema DDE sea Estable Entrada–Estado (ISS)[2]. Esto significa que si el sistema es perturbado con una entrada arbitraria pero acotada, entonces la evolución de todos sus estados también será acotada.

Teorema 4.2. *Considérese la siguiente DDE:*

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{x}(t - \tau_1(\mathbf{x}, t)), t) \tag{4.17}$$

donde la función $\mathbf{f}$ *es localmente Lipschitz en los primeros dos argumentos y continua a tramos en* t*, y* $\tau_1(\cdot)$ *es localmente Lipschitz en* $\mathbf{x}$ *y continua a tramos en* t*. Sea* $\phi_a(t)$ *su solución analítica para una historia inicial* $\phi_a(t < 0)$ *dada. Supóngase que la solución analítica es acotada, y que el sistema forzado*

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{x}(t - \tau_1(\mathbf{x}, t)), t) + \mathbf{u}(t) \tag{4.18}$$

es ISS para la solución $\phi_a(t)$*.*

Sea $\phi(t)$ *la solución de la aproximación DQSS*

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{q}(t), \mathbf{q}(t - \tau_1(\mathbf{q}, t)), t) \tag{4.19}$$

con historia inicial $\phi(t < 0) = \phi_a(t < 0)$*.*

Luego, considerando quantums ΔQ_j*,* $j = 1, \ldots, n$ *suficientemente pequeños, el error* $\mathbf{e}(t) \triangleq \phi(t) - \phi_a(t)$ *estará acotado para todo* $t > 0$*. Aún más,* $\mathbf{e}(t) \to 0$ *si* $\Delta Q_j \to 0$*.*

Demostración. Ver Sección A.3 □

Observación: El Teorema 4.2 se cumple para cualquier DDE de la forma Ec.(4.17), que posee una única función de retardo τ_1. Sin embargo, puede ser extendida de forma directa a DDE con múltiples funciones de retardo, de la forma de Ec.(4.5).

4.4. Resultados Experimentales

Se experimentó y midió el desempeño de la solución numérica aplicada a algunos sistemas DDE típicos tomados de la literatura abierta. Se ejecutaron simulaciones aplicando algoritmos basados en DQSS y se comparó los resultados con aquellos obtenidos usando algoritmos basados en la discretización del tiempo (desarrollados y publicados por otros autores [ST01, S.P08]). Los resultados se obtuvieron experimentando en una plataforma de referencia bajo condiciones controladas.

Se utilizó el bloque Delay descrito en la Sección 4.2.3 como un modelo DEVS atómico en la herramienta PowerDEVS, que provee facilidades especialmente diseñadas para simular sistemas híbridos basado en métodos QSS y ofrece una interfaz gráfica de usuario similar a la de Matlab/Simulink. Los bloques DEVS atómicos se programan en C++ y pueden editarse con la herramienta de edición

[2]Ver A.1 para una definición detallada de sistema Estable Entrada–Estado

de PowerDEVS que provee una estructura de código predeterminada a modo de guía para simplificar la tarea.

El bloque Delay es el único modelo nuevo requerido para simular DDE con PowerDEVS, dado que la herramienta ya provee modelos de tipo funciones estáticas e integradores cuantificados con histéresis.

El nuevo bloque Delay se incorporó como parte estándar de PowerDEVS, en la biblioteca de modelos continuos.

4.4.1. Ejemplo 1

Se considera como primer ejemplo al sistema reportado como *Example 1* en [ST01] caracterizado por la DDE en la Ec. (4.20). La DDE será simulada para $t \in [0, 5]$ con historia inicial $x_1(t) = 1$, $x_2(t) = 1$, y $x_3(t) = 1$ para $t \leq 0$.

$$\begin{aligned} \dot{x}_1(t) &= x_1(t-1) \\ \dot{x}_2(t) &= x_1(t-1) + x_2(t-0{,}2) \\ \dot{x}_3(t) &= x_3(t) \end{aligned} \tag{4.20}$$

Este sistema tiene tres variables de estado x_1, x_2, y x_3 y dos retardos constantes $\tau_1 = 1, \tau_2 = 0{,}2$. Se trata de un sistema DDE con historia constante. Para modelar la aproximación DQSS de la Ec. (4.20), se utilizará la interfaz de usuario gráfica orientada a diseño con bloques de PowerDEVS, como se muestra en la Figura 4.7. Puede verse que se necesitan dos bloques Delay, uno para obtener $x_1(t - \tau_1)$, y otro para obtener $x_2(t - \tau_2)$. Los bloques para generación de señales constantes continuas producen las señales de retardos constante, las cuales se conectan al puerto inferior de los bloques Delay.

Los resultados de simulación se muestran en la Figura 4.8.

El sistema es analíticamente inestable, es decir, las trayectorias crecen exponencialmente en el tiempo.

En la Tabla 4.1 se muestran resultados comparativos del desempeño promedio para grupos de 30 simulaciones con DQSS3 (el algoritmo de integración QSS de tercer orden implementado en PowerDEVS) y con dde23 (otro algoritmo de tercer orden, a tiempo discreto, implementado en Matlab). Se aplicaron los mismos valores de tolerancia de error en ambos casos. La tolerancia relativa se fijó en 10^{-3}. Para los valores de variables de estado muy cercanos a cero, $|x_i| < 10^{-6}$, la tolerancia relativa de error se reemplaza por una tolerancia absoluta de 10^{-6}.

Se analizó tanto el número de evaluaciones de funciones ejecutadas como los tiempos de ejecución de la simulación. El número de evaluaciones de funciones no puede ser comparado directamente (uno a uno), debido a que DQSS3 es un método asíncrono, es decir, evalúa funciones para cada variable de estado de forma independiente (cuando cada variable lo requiera) , mientras que dde23 es un algoritmo sincrónico que evalúa todas las funciones, en forma conjunta, para todas las variables de estado (cuando cualquier variable lo requiere).

Consecuentemente, cuando se simula un sistema de tercer orden (tres variables de estado), se requieren tres evaluaciones de funciones *separadas* en DQSS para actualizar todos los estados, mientras que se requiere una *única* evaluación de función para actualizar las variables en dde23. Por esta razón, para los algoritmos dde23 y dde_solver, se multiplicó el número reportado de cantidad de evaluación de funciones por el número del orden del sistema simulado, obteniendo así resultados comparables.

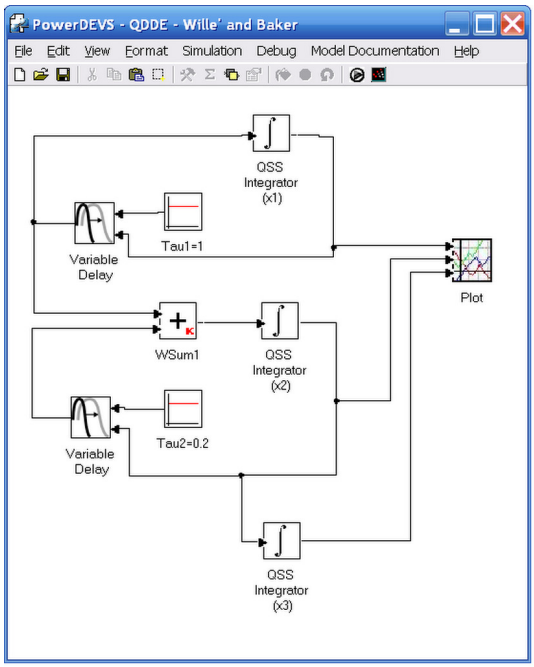


Figura 4.7: Modelo de una DDE (propuesta por Willé y Baker en [WB92]) implementado en la herramienta PowerDEVS.

Ambas herramientas, PowerDEVS y Matlab, realizan compilaciones parciales previas de los algoritmos y/o modelos. Sin embargo, ninguno genera un algoritmo directo que sea simplemente compilado y ejecutado. Luego, ambas herramientas experimentan overheads similares y por ello tiene sentido comparar sus tiempos de ejecución.

Tiempo de Simulación	Método	# de Eval. de Funciones	Tiempo de Ejecución	Speedup
5 s (reported)	DQSS3 dde23	74 264	0.0158 0.0406	2.57
10 s	DQSS3 dde23	162 498	0.0157 0.0484	3.08
15 s	DQSS3 dde23	246 723	0.0157 0.0557	3.54
20 s	DQSS3 dde23	332 948	0.0155 0.0656	4.24
RelTol:1E-3, AbsTol:1E-6				

Tabla 4.1: Análisis de Speedup para DQSS3 vs. dde23 resolviendo la Ec. (4.20)

En PowerDEVS, el algoritmo de resolución en sí es suficientemente veloz como para que su peso relativo respecto del proceso de inicialización sea des-

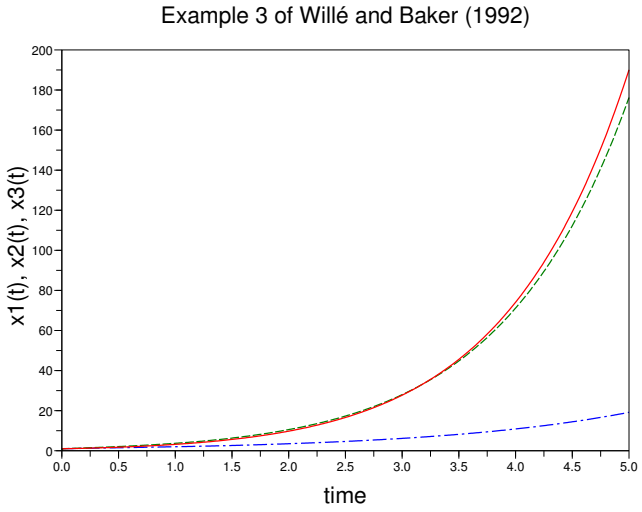


Figura 4.8: Solución de una DDE propuesta por Willé y Baker en [WB92].

preciable (en este ejemplo). Los 0.0158 segundos de ejecución reportados corresponden al proceso de inicialización en todos los casos. Sin embargo, dado que el sistema simulado es analíticamente inestable, el tiempo de simulación no puede ser llevado mucho más adelante sin incurrir en problemas de overflow de representación numérica.

Análisis de Error

Hay una diferencia importante entre DQSS y dde23 en el modo en el que ambos interpretan los valores de tolerancia de error. Mientras que dde23 los interpreta como una medida del error de integración *local* en un paso de integración simple, DQSS los interpreta como una medida del error de integración *global* a lo largo de toda la simulación. Por este motivo, es importante comparar ambas soluciones con respecto a una solución de referencia y medir los verdaderos errores de simulación cometidos por los algoritmos.

Para obtener una solución de referencia, se ejecutaron simulaciones usando ambos algoritmos con requerimientos de precisión más exigentes (en tres órdenes de magnitud) que los utilizados en los experimentos originales. Al comparar los resultados obtenidos de este modo, se pudo confirmar que están en concordancia con una precisión que es mucho mejor que aquella usada en las simulaciones originales.

Se reportará ahora los errores absolutos obtenidos, es decir, las desviaciones de las soluciones originales respecto de la solución adoptada como referencia. En la Tabla 4.2 se muestra el error absoluto máximo (Max), y el error cuadrático medio (Rms).

Para interpretar estos resultados, nótese que $x_1(t)$ crece durante toda la simulación hasta un valor de aproximadamente 20. Luego, con una tolerancia

Tiempo de Simulación	Método	# Evaluac. de Funciones	Tiempo de Ejecución	Speedup	Error Rms $\times 10^{-6}$	Error Max $\times 10^{-6}$
5 s	DQSS3†	74	0.0158	2.57	2680.7	9020.5
	dde23†	264	0.0406		27.4	86.0
	DQSS3‡	284	0.0158	2.57	43.0	128.9
	DQSS3*	598	0.0336	1.20	4.4	17.7
Mediciones de error para la variable: x_1 † RelTol:1E-3, AbsTol:1E-6 ‡ RelTol:1E-5, AbsTol:1E-6 * RelTol:1E-6, AbsTol:1E-6						

Tabla 4.2: Análisis de Error y Speedup para DQSS3 vs. dde23 resolviendo la Ec.(4.20)

de error relativo de 10^{-3}, sería aceptable un error máximo absoluto (que ocurre al final de la simulación) del orden de $20 \cdot 10^{-3}$. DQSS3 produjo un error máximo de $9 \cdot 10^{-3}$. Por razones que no pudieron establecerse, dde23 produjo, para este ejemplo, un error absoluto mucho menor que el requerido.

Con el fin de comparar mejor ambos algoritmos, se procedió a disminuir las tolerancias de error de DQSS3 hasta 10^{-5} e incluso 10^{-6}. Con una tolerancia de error de 10^{-5}, se aceptaría un error absoluto de $20 \cdot 10^{-5}$. DQSS3 produjo un error máximo de $12{,}89 \cdot 10^{-5}$. Con una tolerancia de error de 10^{-6}, se aceptaría un error absoluto de $20 \cdot 10^{-6}$. DQSS3 produjo un error máximo de $17{,}7 \cdot 10^{-6}$.

Por lo anterior, puede verse que DQSS3 ejecuta simulaciones que son tan precisas como se le requiere, pero no mucho más precisas. En todas las simulaciones DQSS3 resultó ser más veloz que dde23.

4.4.2. Ejemplo 2

En este ejemplo se estudia el sistema escalar reportado como *Example 3* en [ST01], cuya DDE queda definida por la Ec. (4.21). El modelo se simulará para $t \in [0, 20]$ con historia inicial $x(t) = t$ para $t \leq 0$.

$$\dot{x}(t) = -\lambda \cdot x(t-1) \cdot (1 + x(t)) \tag{4.21}$$

Este sistema tiene un retardo simple constante $\tau_1 = 1$, es no lineal, y posee una historia inicial variable. Los resultados de la simulación se muestran ene la Figura 4.9.

Los resultados de desempeño en promedio para conjuntos de 30 simulaciones se muestran en la Tabla 4.3.

En este ejemplo, DQSS3 siempre requiere algunas pocas evaluaciones de función más que dde23. Dado que el sistema a ser simulado responde a un modelo de primer orden, el número de evaluaciones de funciones puede ser comparado mejor (uno a uno) entre ambos algoritmos. Entonces, se esperaría que dde23

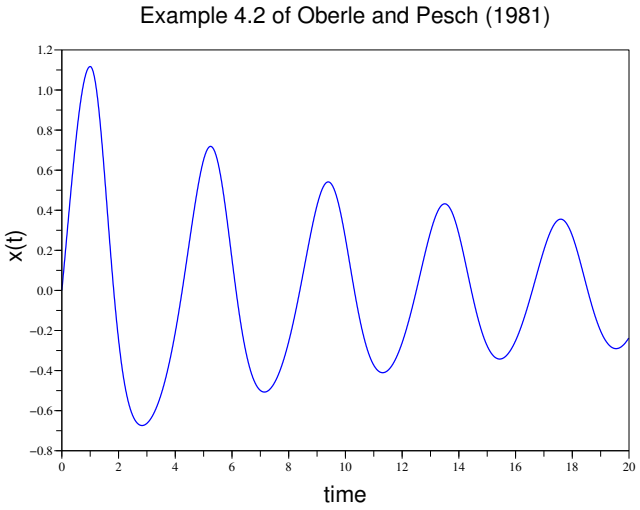


Figura 4.9: Solución de una DDE propuesta por Oberle y Pesh en [OP81] (caso $\lambda = 1{,}5$).

Tiempo de Simulación	Método	# de Eval. de Funciones	Tiempo de Ejecución	Speedup
20 s (reported)	DQSS3 dde23	3825 3679	0.2185 0.9219	4.22
40 s	DQSS3 dde23	7379 7372	0.4359 2.1063	4.83
60 s	DQSS3 dde23	11121 11092	0.6561 3.6891	5.62
80 s	DQSS3 dde23	14931 14800	0.8908 5.5755	6.26
$\lambda = 1{,}5$ RelTol:1E-6, AbsTol:1E-10				

Tabla 4.3: Análisis de Speedup para DQSS3 vs. dde23 resolviendo la Ec. (4.21)

ejecute un levemente más rápido que DQSS3. Aún así, y debido a la naturaleza mucho más sencilla de DQSS3 en comparación con el algoritmo usado por dde23, DQSS3 mejora en velocidad a dde23.

Análisis del Error

Como en el Ejemplo 1 (Sección 4.4.1), se comparará los errores que se obtuvieron utilizando ambos algoritmos. Se usará la misma metodología explicada en la discusión del Ejemplo 1. En este caso se pedirá una tolerancia de error relativo de 10^{-6}. Dado que la solución misma es del orden de 1.0, esto significa que también el error absoluto aceptado debería ser del orden de 10^{-6}.

Tiempo de Simulación	Método	# Evaluac. de Funciones	Tiempo de Ejecución	Speedup	Error Rms $\times 10^{-6}$	Error Max $\times 10^{-6}$
20 s	DQSS3†	3825	0.2185	4.22	0.9	3.2
	dde23†	3679	0.9219		2.7	5.7
	DQSS3‡	8133	0.4378	2.1	0.1	0.5
$\lambda = 1{,}5$ † RelTol:1E-6, AbsTol:1E-10 ‡ RelTol:1E-7, AbsTol:1E-10						

Tabla 4.4: Análisis de Speedup para DQSS3 vs. dde23 resolviendo la Ec. (4.21)

Para comprender los errores de simulación aún mejor, en la Figura 4.10 se grafican los errores absolutos, es decir, las desviaciones en las trayectorias de simulación respecto de la solución de referencia, como funciones del tiempo.

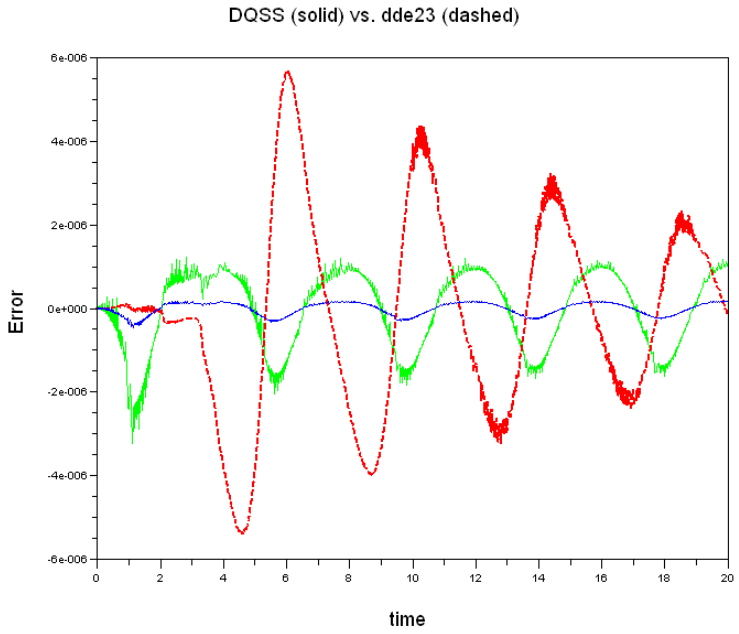


Figura 4.10: Análisis gráfico de errores de simulación (Ejemplo 2)

Los errores más grandes son los cometidos por dde23. El error alcanza un máximo de $5{,}7 \cdot 10^{-6}$ luego de aproximadamente 6 segundos de simulación. El error de simulación cometido por dde23 es aproximadamente proporcional a la solución en sí, es decir, el error es grande y positivo cuando la solución también lo es.

La curva de error intermedio describe los errores cometidos por DQSS3 con un error relativo de 10^{-6}. El error alcanza un máximo absoluto de $3{,}2 \cdot 10^{-6}$ luego de aproximadamente 1 segundo de simulación. Las oscilaciones de los errores están corridas en fase en comparación con aquellas producidas por dde23.

Mientras dde23 parece calcular una solución con una amplitud que es (en promedio) un tanto demasiado grande, DQSS3 parece calcular una solución con una amplitud que es (en promedio) un tanto demasiado pequeña.

Los resultados más precisos se obtienen por DQSS3 con una tolerancia de error relativo de 10^{-7}, reportado en la Tabla 4.4. El error alcanza un máximo absoluto de $5 \cdot 10^{-7}$ luego de aproximadamente 1 segundo de simulación. Las curvas de error para las dos simulaciones de DQSS3 son cualitativamente similares, excepto que los errores de la simulación con más precisión son, en promedio, 9 veces menores.

Para obtener estos resultados más precisos, el número de evaluaciones de funciones tuvo que crecer en un factor de $\frac{8133}{3825} = 2{,}126 \approx \sqrt[3]{10}$, lo cual coincide con lo esperado para un método DQSS con precisión de tercer orden.

4.4.3. Ejemplo 3

Aquí se tratará un sistema DDE un tanto más complicado, reportado como *Example 4* en [ST01], cuya DDE está definida en Ec. (4.22).

$$\begin{aligned} \dot{x}_1(t) &= -x_1(t)x_2(t-1) + x_2(t-10) \\ \dot{x}_2(t) &= x_1(t)x_2(t-1) - x_2(t) \\ \dot{x}_3(t) &= x_2(t) - x_2(t-10) \end{aligned} \tag{4.22}$$

Este modelo DDE debe ser simulado para $t \in [0, 40]$ con historia inicial $x_1(t) = 5$, $x_2(t) = 0{,}1$, y $x_3(t) = 1$ para $t \leq 0$. Se trata de un sistema de tercer orden, no lineal, con múltiples retardos constantes $\tau_1 = 1$ y $\tau_2 = 10$ e historia inicial constante. Los resultados de simulación se presentan en la Figura 4.11.

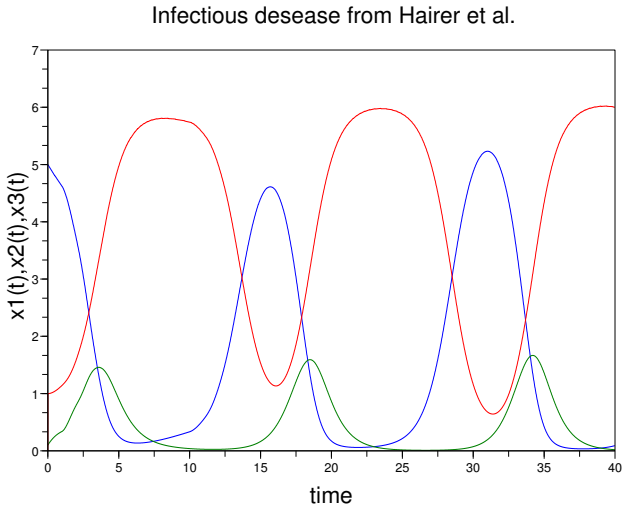


Figura 4.11: Solución de una DDE propuesta por Hairer at al. en [HNW00]

Los resultados de desempeño promedio para conjuntos de 30 simulaciones se muestran en la Tabla 4.5.

Tiempo de Simulación	Método	# de Eval. de Funciones	Tiempo de Ejecución	Speedup
40 s (reported)	DQSS3 dde23	1104 1353	0.0470 0.1734	3.68
80 s	DQSS3 dde23	2328 2568	0.0470 0.3286	6.99
RelTol:1E-3, AbsTol:1E-6				

Tabla 4.5: Análisis de Speedup para DQSS3 vs. dde23 resolviendo la Ec. (4.22)

Al simular este modelo, DQSS3 ejecuta una cantidad levemente menor de evaluaciones de funciones que dde23. El factor de mejora (Speedup) en tiempos de simulación es del orden del obtenido para los ejemplos anteriores.

Análisis del Error

Para este ejemplo se requirió una tolerancia de error relativo de 10^{-3}. Dado que la solución en sí es del orden de 6.0, entonces el error absoluto debería ser del orden de $6 \cdot 10^{-3}$.

Tiempo de Simulación	Método	# Evaluac. de Funciones	Tiempo de Ejecución	Speedup	Error Rms $\times 10^{-3}$	Error Max $\times 10^{-3}$
40 s	DQSS3† dde23† DQSS3‡	1104 1353 703	0.047 0.1734 0.0314	3.68 5.52	1.032 10.701 13.599	6.28 43.75 59.0
Error measurements for variable: x_1 † RelTol:1E-3, AbsTol:1E-6 ‡ RelTol:1E-2, AbsTol:1E-6						

Tabla 4.6: Análisis de Speedup para DQSS3 vs. dde23 resolviendo la Ec. (4.22)

DQSS3 produjo casi de manera exacta el error requerido. Su error absoluto máximo durante los 40 segundos de simulación fue de $6{,}28 \cdot 10^{-3}$. En contraste, dde23 produjo un error absoluto máximo de $43{,}75 \cdot 10^{-3}$, es decir, un error que es aproximadamente siete veces más grande. Aún así, este error es aceptable ya que dde23 controla solo el error local, y es usual esperar que el error global sea un orden de magnitud mayor que el error local.

Con el propósito de comparar mejor ambos algoritmos, se repitieron las simulaciones DQSS3 con una tolerancia relajada en un orden de 10^{-2}. Los nuevos resultados de desempeño se muestran en la Tabla 4.6. Ahora debería esperarse un error de $6 \cdot 10^{-2}$, mientras que DQSS3(rel.tol=10^{-2}) produjo un error de $5{,}9 \cdot 10^{-2}$. Al relajar la tolerancia en un factor de 10, el número de evaluaciones de funciones se redujo sólo en un factor de $\frac{1104}{703} = 1{,}57$, es decir, un tanto menos que $\sqrt[3]{10}$.

Ambas simulaciones DQSS3 fueron más eficientes que la simulación dde23.

Para comprender mejor los errores nuevamente se graficó la serie temporal de errores absolutos, es decir, los desvíos de las trayectorias respecto de la solución referencia, como función del tiempo. En la Figura 4.12 se grafican los errores de dde23 junto con los errores de DQSS3 con los requerimientos de precisión más exigentes.

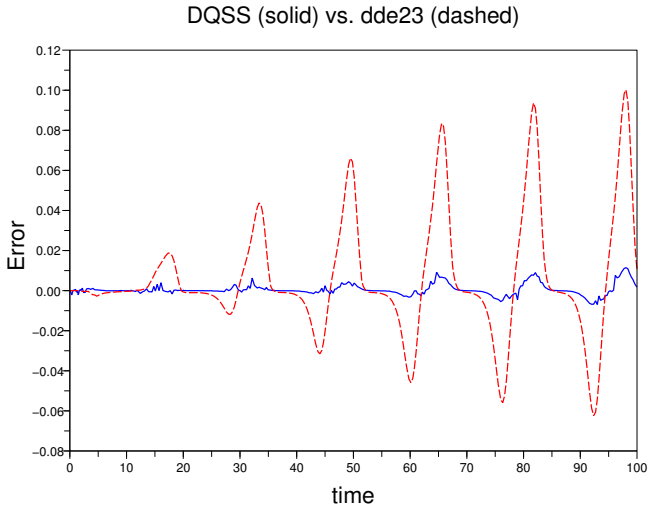


Figura 4.12: Análisis gráfico de errores de simulación (Ejemplo 3)

Las simulaciones parecen ser levemente inestables, a medida que los errores absolutos crecen. Sin embargo, se trata de un efecto artificial. El sistema exhibe un ciclo límite, es decir, es marginalmente estable. La solución numérica acumula a lo largo del tiempo un pequeño error de fase que se manifiesta en las gráficas en forma de errores absolutos crecientes. Este hecho se hace evidente cuando se observan las trayectorias de x_1 en la Figura 4.13. La trayectoria dde23 se adelanta a la (más precisa) DQSS3 por una pequeña cantidad Δt.

4.4.4. Ejemplo 4

En este último ejemplo, se tratará con un sistema DDE mucho más exigente. Se trata de un sistema variante en el tiempo con impulsos, reportado como

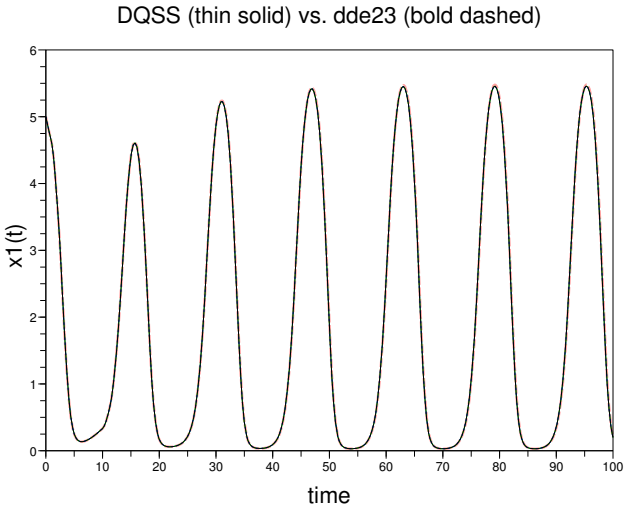


Figura 4.13: Trayectorias de x_1 para el Ejemplo 3

Example 3.1 en [S.P08]. El modelo DDE se define en la Ec. (4.23).

$$\begin{aligned}
\dot{x}_1(t) =& -6x_1(t) + sin(2t)f(x_1(t)) + cos(3t)f(x_2(t)) \\
&+ sin(3t)f\left(x_1\left(t - \frac{1+cos(t)}{2}\right)\right) + sin(t)f\left(x_2\left(t - \frac{1+sin(t)}{2}\right)\right) + 4sin(t) \\
\dot{x}_2(t) =& -7x_2(t) + \frac{cos(t)}{3}f(x_1(t)) + \frac{cos(2t)}{2}f(x_2(t)) \\
&+ cos(t)f\left(x_1\left(t - \frac{1+sin(t)}{2}\right)\right) + cos(2t)f\left(x_2\left(t - \frac{1+sin(t)}{2}\right)\right) + 2cos(t) \\
f(x) =& \frac{|x+1| - |x-1|}{2}
\end{aligned} \tag{4.23}$$

La historia inicial está dada por $x_1(t) = -0{,}5$ y $x_2(t) = 0{,}5$. En cada instante de impulso, $t_k = 2k$, se aplica una solución impulsiva dependiente del tiempo, reemplazando a $x_1(t_k)$ por $1{,}2x_1(t_k)$, y a $x_2(t_k)$ por $1{,}3x_2(t_k)$. La simulación resultante se muestra em la Figura 4.14.

Este ejemplo incluye retardos y discontinuidades en sus estados. El número de evaluaciones de funciones $f()$ en PowerDEVS no incluye el número de veces en los que ambos integradores son reseteados debido a los impulsos. El número total de impulsos es de 52.

Se muestran los resultados de desempeño promedio para 30 simulaciones en la Tabla 4.7.

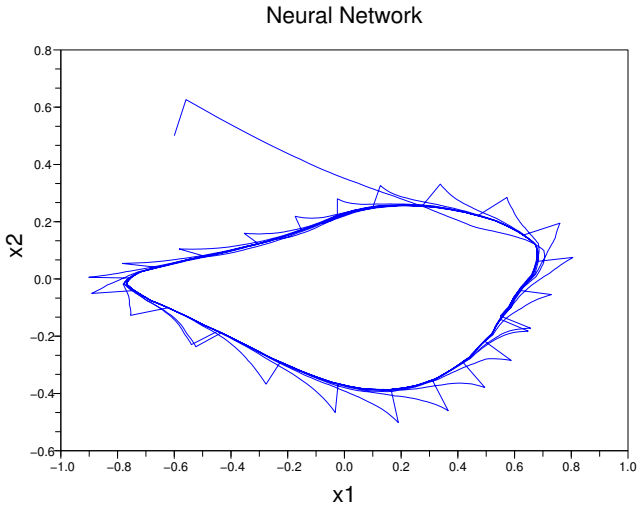


Figura 4.14: Solución de una DDE propuesta por Yongqing y Cao en [YC07]. Retrato de fase de una red neuronal modelada con una DDE con impulsos dependientes del tiempo.

Tiempo de Simulación	Método	# Evaluac. de Funciones	Tiempo de Ejecución	Slowdown	Error Rms $\times 10^{-6}$
50 s	DQSS3†	15003	0.636	4.07	1154.4
	dde_solver†	15822	0.156		9899.1
	DQSS3‡	11512	0.579	3.51	3062.9
Errores de medición para la variable: x_1 † RelTol:1E-3, AbsTol:1E-6 ‡ RelTol:1E-2, AbsTol:1E-6					

Tabla 4.7: Análisis de Speedup y Error para DQSS3 vs. dde_solver resolviendo la Ec. (4.23)

Para este ejemplo, no se pudo disponer del código dde23 para Matlab, sino únicamente el código compilado en Fortran (dde_solver). Por este motivo, los tiempos de ejecución de DQSS3 y dde_solver no son comparables. El código Fortran compilado es más eficiente que el código interpretado de PowerDEVS, lo cual no causa demasiada sorpresa. El número de evaluaciones de funciones producido por DQSS es significativamente menor que el producido por dde_solver.

En este caso se requirió una tolerancia de error relativo de 10^{-3}. Dado que la solución en sí es del orden de 1.0, entonces el error absoluto esperado será del orden de 10^{-3} también.

Análisis del Error

Para este ejemplo, no se provee el error absoluto máximo, sino únicamente los errores medios cuadráticos. La razón es que este ejemplo contiene discon-

tinuidades. Dado que los instantes en los que se ejecutan las discontinuidades varían levemente entre la simulación original y la solución referencia, la gráfica de error en la Figura 4.15 contiene picos que son del orden de magnitud de la señal en sí misma.

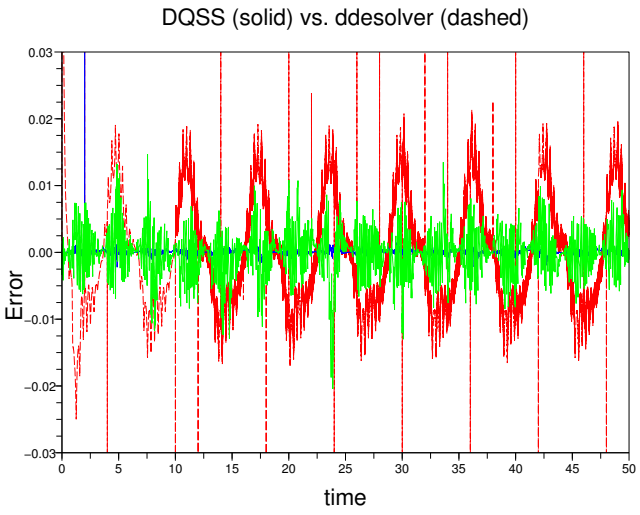


Figura 4.15: Análisis gráfico de errores de simulación (Ejemplo 4)

Observando la Figura 4.15, se ve que los errores absolutos más grandes para DQSS3(rel.tol=10^{-3}) –excluyendo los picos– son aproximadamente del orden de 10^{-3} como se requirió, mientras que los mayores errores absolutos para DQSS3(rel.tol=10^{-2}) son aproximadamente del orden de 10^{-2}. Luego, DQSS3 está calculando soluciones que son casi exactamente tan precisas como se esperaría. Los errores absolutos mas grandes para dde_solver(rel.tol=10^{-3}) son aproximadamente $2 \cdot 10^{-2}$, es decir, cerca de 20 veces mayores que lo requerido. Sin embargo, esto es aceptable considerando que dde_solver controla el error de integración local únicamente, mientras que DQSS3 controla el error de integración global.

4.5. Conclusiones

En el presente capítulo, se presentó una nueva clase de algoritmos de integración para Ecuaciones Diferenciales con Retardo (DDE) y se mostró su aplicación por medio de un número de problemas de la literatura, estudiando su desempeño. La aplicación de la nueva herramienta DQSS a un caso práctico de control de congestión de red será analizada en detalle en la Sección 7.3.1. Sin embargo, como se hizo evidente en el presente capítulo, esta herramienta trasciende el dominio de aplicación de las redes de datos, constituyendo un avance en el estado del arte de la disciplina de modelado y simulación en general.

Los nuevos métodos DQSS se basan en la cuantificación de las variables de estado en lugar de discretizar el tiempo, ofreciendo algunas propiedades relevantes muy interesantes que los hace particularmente atractivos.

En primer lugar, los métodos de Sistemas de Estados Cuantificados (QSS, por *Quantized State Systems*) son naturalmente asíncronos. Se trata de métodos de paso variable, mientras que cada variable de estado se actualiza a su propio ritmo. Las variables de estado con mayores gradientes se actualizan más frecuentemente que aquellas con baja actividad. Por este motivo, los métodos numéricos basados en QSS explotan implícita e intrínsecamente la ralitud de los modelos a ser simulados.

En segundo lugar, se demostró que un sistema DDE lineal invariante en el tiempo y asintóticamente estable con retardos constantes produce una aproximación QSS equivalente que permanece numéricamente estable para cualquier quantum ΔQ. La granularidad de la cuantificación influye en la precisión de los resultados de simulación, pero no afecta a la estabilidad numérica.

En tercer lugar, se demostró que un sistema DDE de tipo Estable Entrada–Estado (ISS) no lineal y variable en el tiempo, con retardos variables (e incluso dependientes de los estados) produce una aproximación QSS que permanece numéricamente estable para cualquier quantum suficientemente pequeño ΔQ. Para cualquier sistema de este tipo, existe un quantum $\Delta Q_{\max}$ mayor que cero tal que su aproximación QSS permanece numéricamente estable para todo $\Delta Q \leq \Delta Q_{\max}$.

En cuarto lugar, se mostró que cuando ΔQ se aproxima a cero en este tipo de sistemas, las trayectorias de simulación convergen a las trayectorias analíticas del sistema DDE original.

En quinto lugar, mientras los algoritmos de control de paso usados en métodos de integración por discretización del tiempo usualmente controlan el error de integración local, los algoritmos basados en cuantificación de estado controlan el error de integración global. Luego, estos últimos proveen mayor robustez que sus competidores de tiempo discreto.

En sexto lugar, se mostró por medio de cuatro problemas de análisis de desempeño distintos que la solución numérica de DDE por cuantificación de estados con frecuencia es más eficiente –de forma significativa– que la estrategia basada en discretización del tiempo. La razón para esta mayor eficiencia radica parcialmente en que DQSS explota naturalmente la ralitud de los modelos a simular, y parcialmente en que los algoritmos utilizados para interpolar los estados retardados son muy sencillos y pueden así ser implementados más eficientemente.

En séptimo lugar, el código de DQSS implementado en PowerDEVS, es mucho más sencillo de utilizar que dde23 y que dde_solver, gracias a la interfaz de usuario gráfica intuitiva que provee PowerDEVS (similar a la provista por Simulink).

Capítulo 5

Simulación Embebida en Tiempo Real Basada en Modelos

Durante décadas la comunidad de ingeniería de software ha perseguido el objetivo de crear métodos formales para el desarrollo de sistemas embebidos, y en particular para aquellos con restricciones de tiempo real. A pesar de los numerosos esfuerzos, la mayoría de los métodos existentes son todavía difíciles de escalar, y requieren de costosos esfuerzos de testeo sin garantizar productos libres de errores. En cambio, la ingeniería de sistemas se ha valido generalmente de técnicas de modelado y simulación para mejorar el desarrollo y obtener productos de alta calidad. El desarrollo basado en modelado y simulación es ampliamente utilizado para las etapas tempranas de un proyecto, pero cuando se evoluciona hacia la implementación en la plataforma de hardware final, usualmente los modelos iniciales son abandonados debido a los drásticos cambios de entorno, de paradigma de programación, de manipulación de datos y de control del avance del tiempo.

En el presente capítulo se propone un enfoque basado en modelado y simulación DEVS para mitigar el problema mencionado en las fases finales de desarrollo de software para sistemas embebidos en tiempo real. La estrategia combina las ventajas de un enfoque práctico con el rigor de un método formal, permitiendo conservar la vigencia de los modelos (utilizados previamente para las fases especificación, simulación y verificación) para su uso en las etapas finales de validación e implementación.

El objetivo es obtener un procedimiento completamente basado en modelos DEVS, los cuales sean a) embebidos en el hardware de destino final, b) ejecutados –simulados– en tiempo real y c) conectados de forma transparente a unidades externas hardware (Hardware–In–the–Loop, HIL) con capacidad de acción y reacción. La estrategia mantiene la continuidad de los modelos DEVS, al encapsular las funcionalidades de interacción con el hardware externo en modelos atómicos especiales (Mappers) capaces de interactuar con interfaces de control (Drivers) propias de cada dispositivo.

Para los objetivos a) y b) se utilizará la herramienta ECD++ basada en DEVS presentada en la Sección 2.4.2, dado su diseño específico para sistemas

embebidos en tiempo real. Sin embargo, esto demandará una tarea de traducción de código, ya que en la presente Tesis se propuso la herramienta PowerDEVS para las fases iniciales de modelado, simulación y verificación. Si bien a nivel formal cualquier modelo DEVS es intercambiable entre herramientas basadas en dicho formalismo, el código que implementa el comportamiento de dichos modelos puede diferir drásticamente entre herramientas.

Para el objetivo c) se realizará una integración de ECD++ con un procesador de red Intel IXP2400 y se desarrollarán bibliotecas de interfaz para la operación tipo HIL entre el simulador y los dispositivos de bajo nivel de dicho procesador. También se proveerá de herramientas visuales avanzadas para simplificar y automatizar el desarrollo de modelos ECD++ y de sus bibliotecas de interfaz.

Adicionalmente se integrarán y preconfigurarán las múltiples herramientas de desarrollo necesarias para la experimentación con ECD++ y la placa de red RadiSys ENP2611 (basada en un IXP2400), creando un laboratorio virtualizado portable que simplifique las tareas de validación e implementación de modelos.

5.1. Antecedentes

Se han propuesto diversas técnicas para alcanzar la continuidad y consistencia de modelos hasta su implementación en sistemas embebidos. Por ejemplo BIP [ABS06] define componentes como la superposición de tres capas: Comportamiento (un conjunto de transiciones), Interacciones (entre transiciones) y Prioridades (para elegir entre transiciones). BIP preserva propiedades durante la composición de modelos y soporta análisis y transformaciones entre límites heterogéneos (temporizado/no temporizado, síncrono/asíncrono, disparado por eventos/disparado por datos). Metropolis es un entorno para diseño de sistemas electrónicos que soporta especificación, simulación, análisis formal y síntesis. Se basa en metamodelos con semántica formal, capturando diseños a alto nivel de abstracción e implementando Modelos de Cómputo (MoC, por *Models of Computation*). Ptolemy II [EJL+03] permite el modelado jerárquico y estructurado de sistemas heterogéneos usando un MoC específico que contempla flujo de datos y flujo de control. ECSL (Embedded Control Systems Language) soporta el desarrollo de controladores distribuidos [BGK+06], incluyendo un entorno de dominio específico para sistemas automotrices (extendiendo la familia Matlab con capacidades para especificación, verificación, planificación, análisis de performance, etc.) SystemC y Esterel son lenguajes de nivel de sistema utilizados para simular y ejecutar modelos, y han sido beneficiados por una creciente adopción en la industria [BS08]. SystemC representa sistemas de hardware y software a diferentes niveles de abstracción, permitiendo elegir el nivel deseado para cada componente. Esterel se utiliza para sintetizar hardware y software por medio de un lenguaje basado en reacción y sentencias de alto nivel para manejar concurrencia. Otros esfuerzos importantes incluyen MoBIES (basado en comunicación de sistemas híbridos multi–agente), OCAPI-XL (focalizado en dispositivos en red con procesadores y aceleradores de hardware en un mismo chip), ForSyDe (basado en modelado de sistemas con MoCs heterogéneos, permitiendo simulación basada en Haskell) y Foresight (usando modelado y simulación basado en limitaciones de recursos, modelado gráfico, y generación de modelos ejecutables).

Una de las técnicas más populares es UML-RT, la cual provee una metodología orientada a objetos. En [HS04] se provee una comparación entre UML-RT y DEVS, en la cual se muestra que a pesar de que el Profile UML-RT especifica tiempo, planificación, y performance usando objetos UML, los mismos no están formalmente definidos.

La teoría DEVS provee una sintaxis y semántica sólidas para la representación de estructura, comportamiento y tiempo, que conducen por sí mismos a un MoC bien definido y sin ambigüedades. Sin embargo, DEVS no fue pensado para el diseño y desarrollo de software. Es fundamental entonces dar soporte para la evolución de los modelos simulables hacia piezas de software equivalentes que puedan operar en la complejidad de los entornos embebidos.

5.2. Integración de ECD++ con un Procesador de Red

El crecimiento exponencial de Internet produjo la aparición de nodos de red inteligentes basados en dispositivos programables. La convergencia de voz y datos respetando la calidad de servicio requiere la capacidad de procesar paquetes tomando decisiones de control complejas y sosteniendo altas velocidades. Aun más, con la rápida evolución de los protocolos, estándares y aplicaciones, la lógica de control de las redes debe adaptarse a un ritmo mucho mayor que el previsto para el recambio tecnológico del hardware subyacente. A su vez, la velocidad de las redes crecen a un ritmo que supera al crecimiento de la velocidad de clock de las CPU en aproximadamente un orden de magnitud, haciendo comparables el tiempo de transmisión de un paquete con el de acceso a una posición de memoria. Esto presenta un desafío tecnológico complejo ya que requiere combinar la flexibilidad de programación de los procesadores de propósito genérico (CPUs tradicionales) con la alta performance de los circuitos de propósito específico (Application Specific Integrated Circuit, ASIC) para transmitir paquetes a velocidad de línea.

Un procesador de red (Network Processor, NP) es un sistema en un chip (System-on-a-Chip - SoC) diseñado para dar solución a estos problemas, concentrando dispositivos heterogéneos en un único circuito integrado: procesadores de propósito general, microcontroladores programables de propósito específico, unidades de switch, unidades criptográficas, controladores de memoria interna y externa, etc. [Com05].

El diseño de los NP hace factible ubicar a ECD++ en el procesador de propósito genérico (que en la mayoría de los casos opera con alguna versión embebida de Linux) y luego valerse de las bibliotecas de desarrollo propias del NP para comunicar a ECD++ con los microcontroladores de propósito específico.

5.2.1. Procesador de Red Intel IXP2400

El procesador Intel IXP2400 de 2.5 Gbps. [Int04] es un ejemplo de NP estructurado en dos niveles de procesamiento: uno de dinámica lenta (*Slow Data Path*) que consiste en un procesador RISC de propósito genérico XScale (ARM V5TE, 600 MHz., 32 bit, llamado *Core* processor), y uno de dinámica rápida (*Fast Data Path*), que consiste en un cluster de 8 microcontroladores RISC programables multithread (600 Mhz., 32 bits, llamados *MicroEngines*, ME). [Car03].

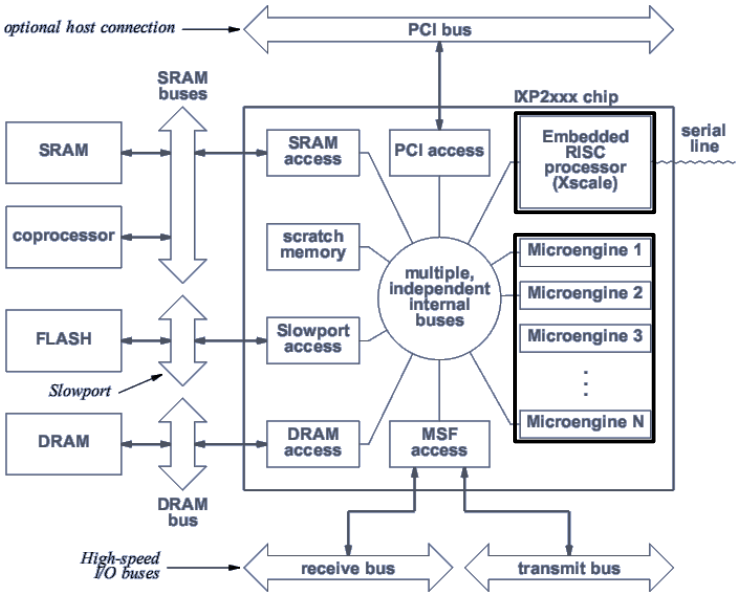


Figura 5.1: Arquitectura de Hardware de la familia IXP2xxx

La arquitectura del IXP2400 permite diseñar motores de reglas flexibles y reconfigurables residentes en el Core, de modo tal que puedan adaptarse dinámicamente sin interferir con la capacidad de las ME de sostener su performance nominal de procesamiento de paquetes [GKS04].

En la Figura 5.1 se muestra un diagrama de bloques de la arquitectura de hardware interna de la familia IXP2xxx. Se destacan a la derecha las dos unidades de procesamiento mencionadas: 1 Core (XScale) y N MicroEngines (con N=8 en el caso del IXP2400).

Los paquetes de red entran y salen vía la MSF (Media Switch Fabric) que provee acceso a los manejadores externos de la capa física de red. Los paquetes son recibidos, transformados y enviados por el código residente en las ME (MicroCode). Sólo ciertos paquetes excepcionales son escalados desde las ME hacia el Core para un tratamiento especial.

Las ME se programan en Assembler o MicroC (una versión adaptada de ANSI C), contando con un set de más de 50 instrucciones que operan a nivel de bit, byte y long–word, se ejecutan en un pipeline de seis etapas y toman en promedio un único ciclo de reloj en finalizar. Las ME poseen 8 threads cada una, y no poseen un sistema operativo. En cambio, se provee un sistema de señales por hardware con las que se administran los context switch entre threads (técnica de “hardware threads”) logrando latencia nula de cambio de contexto. Son estas características, sumadas ciertos registros especiales de acceso rápido local, las que hacen posible el manejo de paquetes a velocidad de línea. Adicionalmente se cuenta con un sistema de marcas de tiempo y temporizadores que ofrecen la capacidad de administrar el tiempo real con una precisión de 16 ciclos de reloj (aproximadamente 0.26 nanosegundos)

Probablemente el aspecto más notable a destacar del IXP2400 sea la organi-

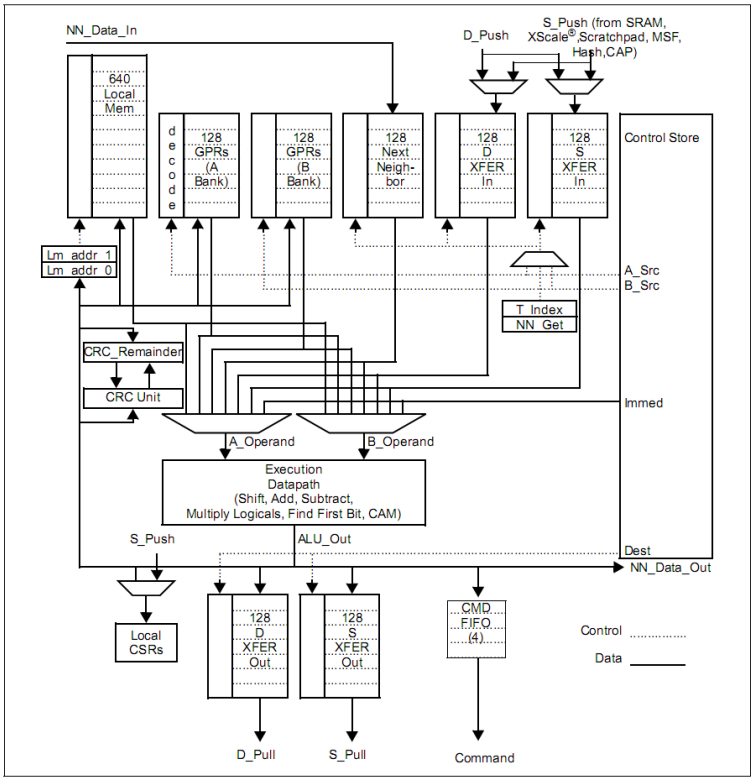


Figura 5.2: Arquitectura de Hardware de una MicroEngine en un procesador IXP2xxx

zación jerárquica de los diversos tipos de memoria (interna y externa) junto con la posibilidad de que la misma sea compartida entre las ME y el Core (donde residirá ECD++).

Los tipos de memoria disponibles son SRAM (8 MBytes), DRAM (256 MByte), Scratchpad (16 KBytes) y Local (2560 bytes por cada ME). SRAM y DRAM son RAMs estática y dinámica respectivamente, externas al chip, y pueden ser accedidas tanto por las ME como por el Core. Scratchpad es una memoria de baja latencia para señalización veloz entre los threads de las ME desde y hacia el Core, operando en forma de interrupciones, datos compartidos y/o estructuras en anillo.

La comunicación interna de baja latencia entre MEs es crítica, y para ello se utiliza la memoria tipo Local que no es accesible por el Core. Incluso existen registros especiales llamados Next Neighbor (NN) que pueden ser accedidos únicamente por 2 ME adyacentes.

En la Figura 5.2 se muestra un diagrama de bloques del hardware interno de una MicroEngine.

Las ME operan mucho más rápido que la memoria externa, completando instrucciones dentro de un mismo ciclo de reloj. Luego, un simple acceso a la memoria RAM podría bloquear la ejecución por varios ciclos de reloj. Comparativamente, mientras acceder a la memoria Local insume 1 ciclo de reloj, acceder

al Scratchpad insume 60 ciclos, a la SRAM 150 ciclos y a la DRAM 300 ciclos.

En síntesis, utilizando memorias y estructuras de datos especializadas y un reemplazo de un sistema operativo por hardware threads, las ME proveen alta performance comparable a la de los ASICs, programabilidad comparable a la de las CPUs, y comunicación con los niveles superiores de dinámica lenta (Core) para lograr administrabilidad, tratamiento de casos excepcionales y flexibilidad ante cambios de políticas de calidad de servicio.

Se trabajará con una placa de red Radisys ENP-2611 [Rad06] como plataforma de desarrollo, un producto comercial basado en el NP IXP2400 que provee conectividad por medio de 3 ports ópticos Gigabit Ethernet.

5.2.2. ECD++ en la Arquitectura de Referencia IXA

El término IXP (Internet Exchange Processor) se refiere a una familia de NPs que implementan la *arquitectura IXA* (Internet eXchange Architecture).

IXA define un marco estándar para desarrollar aplicaciones de red modulares, basadas en bloques de código reusables, reconfigurables, y portables entre distintos NP. La idea básica es definir capas de responsabilidades acotadas y proveer para cada capa bibliotecas estándar accesibles mediante interfaces (Application Program Interface, API) bien definidas. IXA abarca tanto el software que ejecuta en el Core como el de las ME, e Intel provee una gran cantidad de módulos estándar para utilizar a ambos niveles.

Una aplicación de red basada en IXA se compone esencialmente por una cadena de tareas (algoritmos) aplicadas secuencialmente a un flujo de paquetes. Diversas tareas se distribuyen a distintas ME usando diversas estrategias de balanceo de carga. Cada tarea asignada a cada ME posee a su vez un componente de software asociado en el Core, con el cual intercambia información de control, medición y administración.

IXA estructura y ordena la arquitectura de software (bibliotecas) que hacen posible orquestar esta multiplicidad de piezas de código ejecutando en modo paralelo y asíncrono. La Fig. 5.3 muestra la relación entre la arquitectura de hardware discutida en 5.2.1 y la arquitectura de software IXA.

De todas las bibliotecas disponibles, las de mayor interés para lograr la integración de ECD++ son: *Core Components (CC)*, *Resource Manager (RM)*, y *MicroBlocks (MB)*. Las API de CC permiten crear módulos de kernel Linux en el procesador Core, que a su vez utilizan las API de manejadores de recursos (RM) para acceder a estructuras de memoria compartidas entre el Core y las ME. Sin embargo, una misma posición de memoria física es referenciada de modo completamente distinto desde el Core y desde las ME. Por ello, las ME utilizan las bibliotecas MB para acceder a dicha memoria compartida utilizando una estructura de punteros propia distinta a la utilizada desde el Core.

Sin embargo, gracias a IXA se hacen totalmente transparente para los programadores de ambos niveles de código los intrincados detalles específicos de comunicación entre los dos tipos de procesadores que coexisten en el IXP2400. Finalmente, cualquier nueva funcionalidad desarrollada para el Core y/o las ME será reusable y portable a cualquier procesador que se ajuste a la arquitectura IXA.

Como puede observarse en la Figura 5.3 ECD++ se ubica en el Slow Data Path. Se desarrolló una biblioteca llamada *ECD++ I/O Core*, consistente en un *Core Component* que es invocado por ECD++ para comunicar modelos

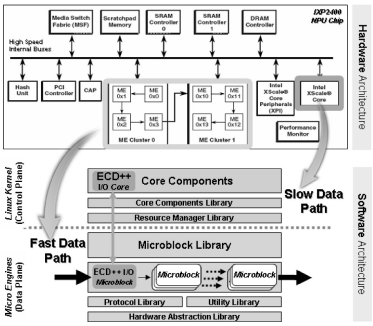


Figura 5.3: ECD++ embebido en un Intel IXP2400 NP.

DEVS con el código que implementa los protocolos de red en el Fast Data Path. También se desarrolló una biblioteca *ECD++ I/O microblock* al nivel MicroEngines para comunicarse con los modelos DEVS en el Core.

Con esta nueva infraestructura de software, se puede reemplazar un subsistema de modelos DEVS que representa a una red real por puertos de comunicación desde y hacia el sistema real de procesamiento de paquetes (Fast Data Path).

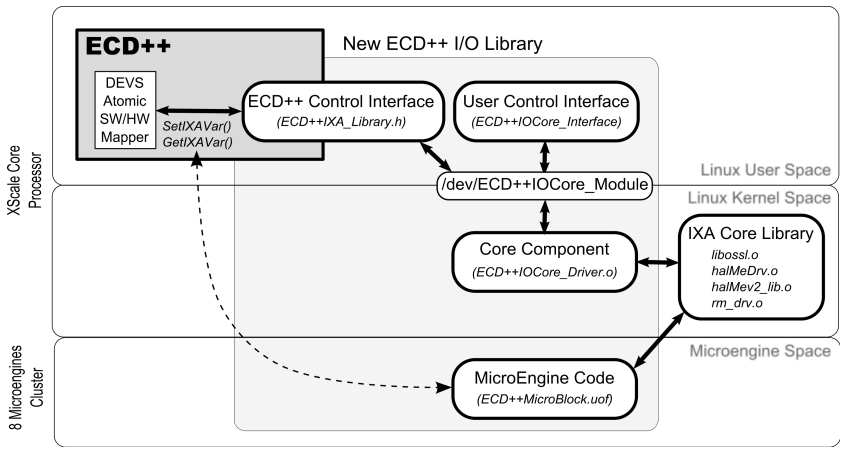


Figura 5.4: Nuevas bibliotecas ECD++ Input/Output para IXA.

5.3. Implementación

Se describirá aquí la adaptación del entorno de modelado y simulación en tiempo real ECD++ haciéndolo capaz de ejecutar en el Core de un NP. También se desarrollaron nuevas bibliotecas para poder comunicar los modelos DEVS con las capas de hardware específicas para manejo de tráfico. Así, los modelos DEVS embebidos controlan al hardware que maneja el tráfico de red, logrando un funcionamiento de tipo Hardware-in-the-Loop (HIL).

5.3.1. Biblioteca de Interfaz

Las bibliotecas de interfaz desarrolladas permiten a cualquier modelo atómico de ECD++ acceder a variables en el espacio compartido con las ME por medio de funciones que se denominarán SetIXAVar() y GetIXAVar(). Cada modelo atómico que desee interactuar con las ME deberá cumplir con tres requisitos:

1. **Declarar** como *Variable IXA* a un parámetro del modelo atómico.

   Esta declaración se realiza en el archivo de acoplamiento .ma que ECD++ utiliza para especificar la estructura del sistema acoplado completo, junto a los parámetros de inicio de cada modelo DEVS atómico.

2. **Incluir** la nueva biblioteca *ECD++IXA_Library.h* en el archivo *.cpp* del modelo atómico.

   Esto permite invocar las funciones SetIXAVar(*Variable IXA*,...) y GetIXAVar(*Variable IXA*,...).

3. **Tener asociado** un bloque de microcódigo, que ejecutará en espacio de Microengines, el cual **importe la variable** *Variable IXA* desde el espacio del Core.

La comunicación se resuelve de modo transparente para el diseñador del modelo atómico, por medio de los módulos de la biblioteca *ECD++ I/O Library*

para IXA que se muestran en la Figura 5.4. En dicha figura, un modelo atómico DEVS llamado *SW/HW Mapper* utiliza la infraestructura de comunicación desarrollada para compartir variables con el microcódigo *ECD++MicroBlock*. De este modo, pueden desarrollarse modelos tipo Mapper que manejen el acceso a distintas variables externas compartidas con IXA, encapsulando dicha funcionalidad, y separándola de la especificación dinámica de los modelos que definen el resto del sistema.

También se diseñó una interfaz de acceso a las *Variables IXA* por línea de comando llamada *ECD++IOCore_Interface*, con propósitos de monitoreo y/o ensayo manual. Esta interfaz puede ser utilizada simultáneamente a la ejecución ECD++, operando sobre las mismas variables.

5.3.2. Generación Automática de Interfaces

Dado que las interfaces de comunicación con IXA proveen servicio a un conjunto de *Variables IXA* definibles por el usuario, las bibliotecas deben regenerarse conforme se modifica la elección de dichas variables (agregado, remoción o renombrado). Acorde a esta necesidad, se diseñó un mecanismo para la generación parametrizable de bibliotecas, capaz de detectar automáticamente las *Variables IXA* elegidas por el modelador (interpretando el archivo de acoplamiento *.ma*)

Como se muestra en la Figura 5.5 se requieren 5 pasos desde la elección del nombre de una Variable IXA, hasta la obtención de toda la infraestructura de bibliotecas necesarias para hacerla accesible desde un modelo atómico DEVS.

Los pasos ① y ② conciernen exclusivamente a ECD++, y fueron mencionados en la sección anterior como el primer y segundo requisito para obtener un modelo DEVS con capacidad de acceso a *Variables IXA*. Estos son pasos sencillos que forman parte de la práctica usual de modelado con ECD++ y pueden resolverse con un editor de texto.

Los pasos ③ y ④ implican generación automática de bibliotecas, y requieren del uso de las nuevas herramientas desarrolladas, consistentes principalmente en scripts Perl.

El paso ⑤ es de exclusiva injerencia de ECD++, es decir, forma parte del proceso usual de compilación del simulador, independientemente del uso o no de *Variables IXA* en algún modelo atómico. Sin embargo, el binario generado será particular para la arquitectura de hardware del procesador embebido de destino (en este caso, IXP2400).

En este procedimiento se asumirá que siempre se dispone de un microcódigo para ejecutar en las Microengines, el cual declara explícitamente su intención de compartir las *Variables IXA* con el espacio de Core. Las características técnicas del microcódigo no serán descritas en esta Tesis por escapar al interés central del trabajo desarrollado.

Este mecanismo automatiza y facilita enormemente la tarea de declaración de variables entre ECD++ y las Microengines, ocultando un gran número de detalles de implementación de bajo nivel. En el ejemplo siguiente se mostrará el resultado de aplicar los pasos mencionados a un caso real con el procesador IXP2400.

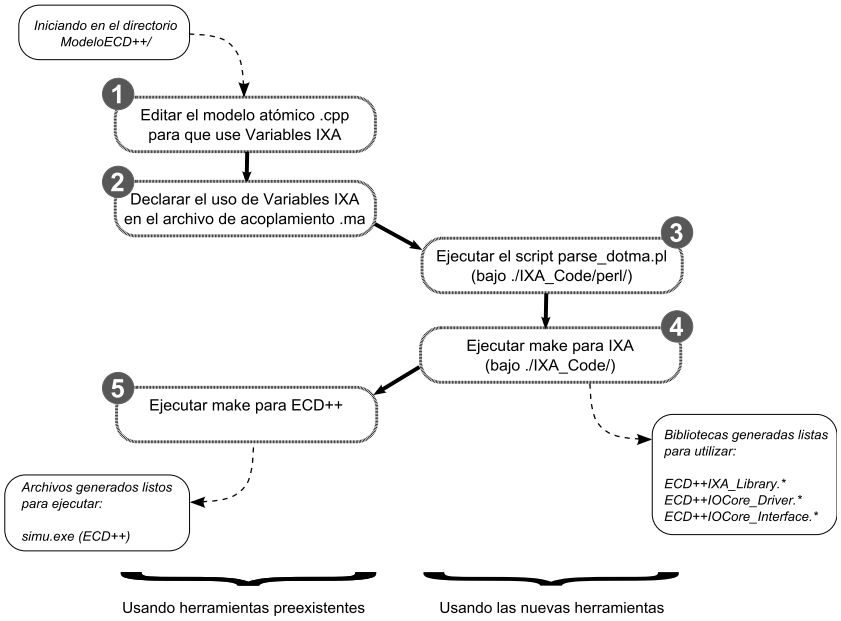


Figura 5.5: Generación automática de las bibliotecas de interfaz ECD++IXA* y ECD++IOCore*

5.3.3. Ejemplo 1. Contador de Eventos

Se mostrará la aplicación del procedimiento explicado hasta aquí a un ejemplo práctico sencillo, emulando un monitor de eventos de tráfico. Se plantea la necesidad de manejar dos *Variables IXA*desde ECD++: una que provea información acerca de la cantidad de ocurrencias de ciertos eventos de tráfico de interés, y otra que provea la capacidad de reiniciar dicho contador. Se las llamará *EventCounter* y *ResetCommand* respectivamente. EventCounter será actualizada por las ME, conforme ocurran los eventos que se pretende monitorear. Cada envío de ResetCommand, además de reiniciar EventCounter a cero, podrá transportar un código numérico para ser interpretado como un comando por el bloque de microcódigo *ECD++MicroBlock* (con el cual interacciona ECD++). El microcódigo deberá incrementar en uno a EventCounter cada vez que acontezca un evento particular. A los efectos de validación de este ejemplo, estos eventos se forzarán artificialmente desde las ME, es decir, no estarán asociados a acontecimientos de tráfico real de red (como podría ser, por ejemplo, la llegada de un paquete de red corrupto).

El archivo de acoplamiento (.ma) se muestra en el Listado 5.1.

```
1 [top]
2 components : eventCounter@trafficEventCounter
3 out : counterStatus
4 Link : counter@eventCounter counterStatus
5 [eventCounter]
6 pollingPeriod : 00:00:02:000
7 counterLimit  : 30000000
```

```
8 EventCounter   : 0 %%useIXAVar[0]
9 ResetCommand   : 7 %%useIXAVar[0]
```

Listing 5.1: Modelo de Acoplamiento IXA_EventCounter.ma

El mismo consta de un único modelo atómico *eventCounter* (una instancia particular del *tipo* de modelo trafficEventCounter). Posee un puerto de salida llamado *counter*, el cual emite valores hacia otro puerto de salida, esta vez del modelo acoplado *top* (el de mayor jerarquía del sistema). El atómico eventCounter se parametriza con 4 variables que afectan su dinámica, dos de las cuales se declaran como *Variables IXA* utilizando el comentario *useIXAVar[0]* indicando que se compartirán con un microcódigo ejecutando en la MicroEngine 0. Este modelo consultará cada pollingPeriod unidades de tiempo a la Microengine por el valor actualizado de la cantidad de eventos de tráfico acontecidos. El valor es consultado mediante la Variable IXA EventCounter, que se inicializa con valor 0. El parámetro counterLimit (inicializado en 30000000) indica que cuando EventCounter supere dicho valor, el modelo deberá enviar un comando de reinicio de contador ResetCommand con valor 7. Dicho comando será interpretado por el microcódigo que recomenzará la cuenta de eventos desde 0.

En el Listado 5.2 se muestran las secciones y líneas de código principales de trafficEventCounter.cpp relacionadas con el uso de *Variables IXA* y con el comportamiento esperado descrito más arriba.

```
1  #include "trafficEventCounter.h"
2  #include "ECD++IXA_Library.h"
3  ...
4  // *** Constructor
5  trafficEventCounter::trafficEventCounter(...):Atomic(
       name),counter(addOutputPort("counter")){}
6  // *** Model Initialization ***
7  Model &trafficEventCounter::initFunction() {
8  ...
9   this->EventCounter = str2Int( MainSimulator::Instance
       ().getParameter( description(), "EventCounter" ) )
        ; // IXA Varaible
10  this->ResetCommand = str2Int( MainSimulator::Instance
       ().getParameter( description(), "ResetCommand" ) )
        ; // IXA Varaible
11 ...
12  this->state = Sleeping;
13  holdIn(active, this->pollingPeriod); // Programs
       itself for the first poll
14 ...}
15 // *** DEVS External Transition Function ***
16 Model &trafficEventCounter::externalFunction(...) {
17 ...}
18 // *** DEVS Internal Transition Function ***
19 Model &trafficEventCounter::internalFunction(...) {
20  switch (this->state) {
21   case Sleeping:
22 ...
```

```
23    this->currentCounter = GetIXAVar("EventCounter");
24    this->state = Notifying;
25    holdIn( active, 0); // Programs itself for emmiting
          inmediately the obtained data
26  case Notifying:
27 ...
28   if ( this->currentCounter > this->counterLimit ) {
29    SetIXAVar("ResetCommand", strResetCommand);}
30   this->state = Sleeping;
31   holdIn(active, this->pollingPeriod); // Programs
        itself for the next poll
32 ...}
33 // *** DEVS Output Function ***
34 Model &trafficEventCounter::outputFunction(...) {
35  switch (this->state) {
36 ...
37   case Notifying:
38    sendOutput(msg.time(), counter, this->
         currentCounter); // Emit the observed
         information
39 ...}
```

Listing 5.2: Definición del modelo atómico trafficEventCounter.cpp

La biblioteca incluida en la fila 2 provee la interfaz de acceso a SetIXAVar() y GetIXAVar(). En la fila 5 se declara el puerto de salida *counter* por el cual se emitirán los valores leídos desde la Microengine. Durante la inicialización del modelo, en las filas 9 y 10 se toman desde IXA_EventCounter.ma los valores iniciales para las *Variables IXA*. El modelo puede estar en dos estados definidos por *state*: Sleeping y Notifying. En la fase Sleeping el modelo espera a que se cumpla el tiempo pollingPeriod hasta enviar la próxima consulta de EventCounter. En la fila 12 el modelo se inicia en Sleeping, y en la fila 13 lo hace esperando pollingPeriod unidades de tiempo. Cuando se agota el tiempo de espera en estado Sleeping, en la fila 23 se lee el nuevo valor del contador con **GetIXAVar**("EventCounter") y en la fila 24 se cambia el estado a Notifying. En la fila 25 se fuerza una transición interna inmediata para poder ejecutar la función de salida y así emitir por el puerto eventCounter el valor almacenado en currentCounter. En la fase Notifying (de duración cero, un estado transitorio) primero se emite el último valor del contador (fila 38, función outputFunction) y luego, en la transición interna (fila 28), se decide si hay que enviar o no un comando de reset. Si es así, en la fila 29 se invoca **SetIXAVar**("ResetCommand",strResetCommand) lo cual escribe la variable IXA ResetCommand haciendo que el microcódigo reaccione al cambio. El valor de strResetCommand deberá ser 7 acorde a lo especificado en el archivo IXA_EventCounter.ma, e importado al modelo atómico en la fila 10. Luego en las filas 30 y 31 se cambia nuevamente al estado Sleeping durante pollingPeriod unidades de tiempo, recomenzando el ciclo. Este modelo no posee puertos de entrada, por lo cual no hace uso de la función de transición externa.

El resultado de la ejecución embebida y en tiempo real del sistema IXA_EventCounter se muestra en la Figura 5.6.

Se muestra un lapso de 10 segundos de ejecución, en los cuales el modelo

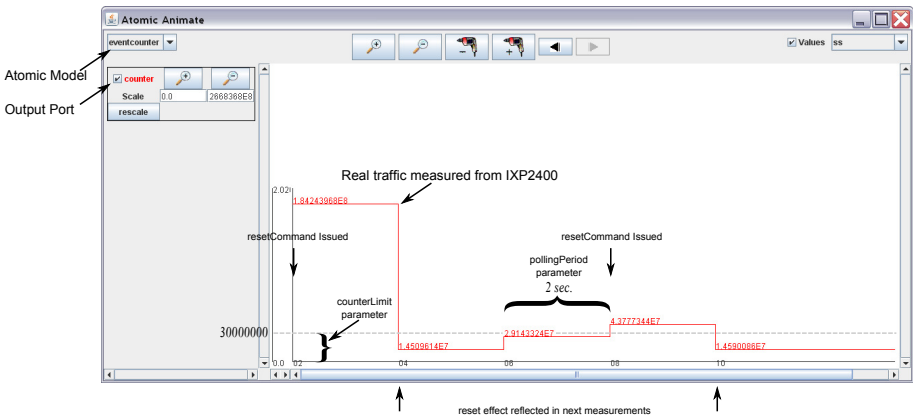


Figura 5.6: Resultado de Ejecución de un Modelos ECD++ en IXP2400. Interacción con MicroEngines en Tiempo Real.

atómico eventCounter detecta en 2 oportunidades que el contador de eventos de las MicroEngines supera el límite establecido por counterLimit. En dichas oportunidades el sistema envía un comando de reinicio, lo cual se refleja en una caída del contador por debajo del umbral en el período de medición subsiguiente. Se valida así el modelo y su interacción efectiva con el hardware especializado de bajo nivel.

5.4. Herramientas y Métodos Avanzados

En esta sección se presentarán nuevas herramientas avanzadas para implementar modelos DEVS embebidos en tiempo real utilizando ECD++. El objetivo particular es obtener herramientas prácticas que simplifiquen el proceso de diseño de sistemas embebidos para control de redes. Se mostrará una nueva interfaz visual para modelado y simulación con ECD++ y se implementará un laboratorio virtualizado y portable para desarrollar modelos DEVS y embeberlos en la placa de red RadiSys ENP-2611 basada en IXP2400.

5.4.1. Herramienta Visual de Modelado Basada en Eclipse

En ECD++ los modelos acoplados se definen mediante un lenguaje declarativo de alto nivel, mientras que los modelos atómicos pueden definirse tanto declarativamente utilizando DEVS-Graphs [1] como programáticamente utilizando C++. Algunas herramientas previas de soporte para ECD++ permiten crear modelos DEVS acoplados y atómicos gráficamente y visualizar con animaciones los resultados de una simulación [Wai09], pero poseen varias limitaciones: son desarrollos independientes que no utilizan interfaces estándar, no poseen mecanismos para la distribución de sus actualizaciones, son difíciles de extender,

[1] En este trabajo se hará referencia exclusivamente a modelos atómicos en C++. Los modelos DEVS-Graphs se basan en grafos para definir los estados y sus transiciones. Cada DEVS-Graphs define los cambios de estado a partir de transiciones internas y externas, y cada una es traducida a una definición analítica [Wai09]

están desacopladas del entorno de simulación y utilizan formatos propios para representar los modelos. Esto dificulta visualizar modelos ECD++ preexistentes, requiriendo cambios de formato y haciendo difícil mantener consistente la representación gráfica con la definición ejecutable de los modelos. Las herramientas previas presentan también varias falencias de usabilidad general, como falta de comandos útiles, acciones poco intuitivas, dificultad para navegar los modelos acoplados jerárquicamente, etc.

Acorde a las dificultades planteadas, se desarrolló la herramienta avanzada CD++Builder 2.0 [BWC10a], un plugin para Eclipse [BBM03] que resuelve los problemas anteriores, basado en el simulador ECD++. La integración de varias herramientas asociadas a ECD++ en un mismo entorno reduce la curva de aprendizaje a nuevos usuarios y simplifica el proceso de modelado y simulación para sistemas embebidos evitando la necesidad de utilizar varias aplicaciones. Los nuevos editores gráficos de CD++Builder para modelos acoplados y modelos atómicos resuelven los problemas de usabilidad a través de asistentes estándar de Eclipse, comandos de copiar y pegar, teclas de acceso rápido, zoom in–out de la vista del modelo, etc. Para los modelos atómicos basados en C++, los editores proveen una representación gráfica de la interfaz externa (nombre del modelo, puertos y parámetros) de los mismos. La representación gráfica y la representación declarativa de modelos ECD++ se mantienen consistentes automáticamente, permitiendo a los usuarios actualizar ambas vistas de modo transparente. Con las nuevas características desarrolladas para los editores, es posible abrir definiciones de modelos realizadas con versiones anteriores de la herramienta y obtener automáticamente una representación gráfica del modelo. Esto es muy beneficioso ya que ECD++ cuenta con una vasta biblioteca de modelos previamente implementados y testeados, pero que no contaban con una representación gráfica que simplifique la comprensión de su estructura y funcionamiento.

La posibilidad de crear y editar modelos atómicos y acoplados gráficamente permite a los usuarios especificar en DEVS modelos complejos sin programar, reusando bibliotecas preexistentes. Cuando es necesario desarrollar nuevos modelos atómicos en C++, se proveen plantillas de código para evitar tareas repetitivas y propensas a errores. Las plantillas poseen estructuras de código estándar y comentarios que promueven buenas prácticas de programación y modelado. La integración con el plugin para Eclipse C/C++ Development Tools (CDT) [Lee04] provee asistencia para el desarrollo en C++ (coloreo de código, resumen de métodos, etc.). Para el análisis de los resultados de las simulaciones, se muestran animaciones de los envíos de mensajes entre modelos y los valores de los puertos de entrada y salida a cada instante. Las animaciones pueden ser lanzadas desde la interfaz de Eclipse. Al utilizar las nuevas representaciones gráficas, es posible visualizar los resultados de modelos acoplados compuestos por modelos atómicos C++, lo cual no era posible anteriormente.

En la Figura 5.7 se presenta una captura de pantalla del nuevo entorno de modelado y simulación resaltando las principales funcionalidades mencionadas hasta aquí.

Adicionalmente, dado que Eclipse es multiplataforma, CD++Builder permite desarrollar modelos DEVS tanto en sistemas Linux como Windows. Por otro lado, es una plataforma estándar para desarrollo de IDEs de vasta adopción internacional, proveyendo una interfaz conocida y fácil de comprender para nuevos usuarios de ECD++. Eclipse provee un framework pensado para ser ex-

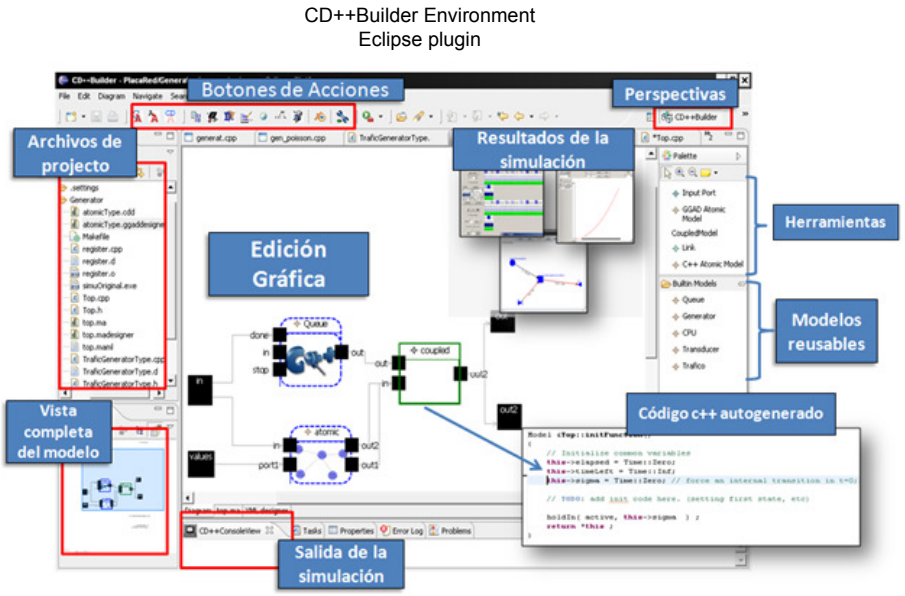


Figura 5.7: Entorno de Simulación CD++Builder para desarrollo de sistemas embebidos con ECD++.

tensible, lo que facilita añadir nuevas herramientas para trabajar con modelos DEVS integrándolas con las preexistentes, dentro de la misma plataforma.

Eclipse también fue extendido para proveer un sitio de actualizaciones, que permite instalar y descargar actualizaciones para los plugins de un único punto accesible a todos los usuarios, facilitando la distribución de nuevas herramientas y soluciones a bugs encontrados. Por otro lado el desarrollo de CD++Builder incluye tests automáticos de regresión, que facilitan la manutención y extensibilidad del plugin, ya que pueden ser utilizados para comprobar el correcto funcionamiento de las funcionalidades ya desarrolladas luego de introducir modificaciones.

La nueva herramienta presentada será utilizada para diseñar, ensayar y visualizar resultados en el Ejemplo 2 de la Sección 5.4.3 que presenta un control supervisorio de calidad de servicio implementado con ECD++.

5.4.2. Laboratorio Virtualizado

La experimentación con el procesador IXP2400 y la arquitectura de referencia IXA ofrece una gran potencia y flexibilidad para desarrollar aplicaciones embebidas para control de red. Sin embargo, la instalación y puesta a punto de las numerosas herramientas y bibliotecas necesarias para obtener un laboratorio productivo consisten en tareas tediosas, muy propensas a fallas, requiriendo en muchos casos de la asistencia de expertos. La operatoria completa puede insumir semanas de trabajo. Esta situación atenta contra la replicación del entorno de experimentación y su adopción por parte de ingenieros, docentes y alumnos que deseen desarrollar aplicaciones basadas en modelos DEVS para ese escenario.

En la Figura 5.8 se muestra un esquema que combina información lógica y

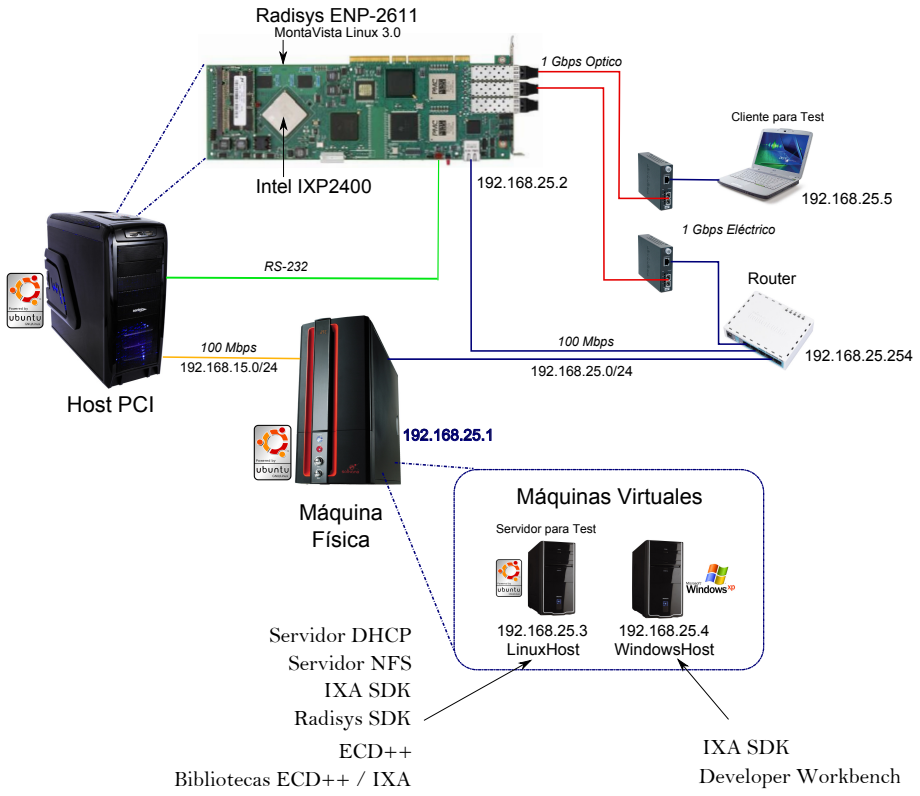


Figura 5.8: Diagrama Lógico–Físico Conceptual: Laboratorio basado en Máquinas Virtuales para desarrollo con ECD++ e IXP2400 incluyendo las nuevas bibliotecas de integración.

física del nuevo laboratorio de referencia desarrollado y de los componentes de software que se ubican en cada nodo. Los componentes de software provienen básicamente de los provistos por Intel y RadiSys, involucrando tres sistemas operativos: Linux Montavista Embedded, Linux de propósito general y Windows. La instalación y ajuste de parámetros se basa en múltiples paquetes de software provistos por los fabricantes, representando un proceso complicado.

Como solución, se realizaron todas las instalaciones y configuraciones necesarias en *máquinas virtuales* portables fácilmente. Es decir, LinuxHOST y WindowsHOST son máquinas virtuales que ejecutan en una misma Máquina Física. En otra máquina física separada (Host PCI) se enchufa la placa RadiSys ENP-2611 en un slot PCI [2] , la cual es administrada vía una subred dedicada y una terminal en modo texto vía RS232. Esto genera un laboratorio preinstalado y totalmente replicable, eliminando prácticamente todo esfuerzo preliminar a desarrollar modelos ECD++ para IXP2400 con la herramienta CD++Builder.

La necesidad de WindowsHOST proviene del hecho de que el desarrollo de microcódigo para las MicroEngines de IXP2400 se realiza con un entorno avan-

[2]Si bien podría insertarse la placa en la misma Máquina Física del laboratorio, esta disposición permite a varios laboratorios virtuales acceder a la placa RadiSys para distintos proyectos y propósitos.

zado de desarrollo y depuración provisto por Intel (Intel Developer Workbench) que funciona bajo Windows XP.

Como se observa en la figura, la máquina virtual LinuxHOST incluye a ECD++ y a las nuevas herramientas presentadas en esta Tesis: las bibliotecas ECD++/IXA y el entorno gráfico avanzado CD++Builder.

Cuando se completa el diseño de un sistema embebido de control basado en DEVS (obtenido usando CD++Builder en LinuxHOST), y ya se dispone del microcódigo que manejará los paquetes reales de red (obtenido usando Developer Workbench en WindowsHOST), ambos archivos binarios son "bajados" al procesador IXP2400 por medio de nuevos scripts de automatización de tareas. Luego, se está en condiciones de realizar ensayos en tiempo real y analizar los resultados por medio de los archivos de log generados por ECD++ en Linux Montavista, que son accesibles vía el Sevidor NFS montado en LinuxHOST.

El ejemplo práctico que se presenta a continuación fue desarrollado íntegramente utilizando el nuevo laboratorio virtual presentado en esta sección.

5.4.3. Ejemplo 2. Control Supervisorio de Calidad de Servicio basado en Medición de Tasa

Como caso de estudio de la metodología de simulación embebida basada en modelos se diseña un sistema de Control Supervisorio de Calidad de Servicio (QoS), el cual acepta políticas globales configurables y monitorea la tasa de tráfico (Traffic Rate – TR) de un flujo real de paquetes. Este control es del tipo sugerido en el escenario de estudio motivador para operar en un nodo enrutador como se presentó en la Figura 2.11.

Dependiendo de las políticas globales y del estado de TR, el sistema envía información de control actualizada a los algoritmos de bajo nivel que deciden y ejecutan el descarte de paquetes. La información de monitoreo y control desde y hacia estos algoritmos de bajo nivel se realiza a través de puertos de entrada–salida de ECD++. La arquitectura IXP2400 permite construir motores de reglas flexibles y reconfigurables residentes en el NP, que pueden ser adaptadas dinámicamente sin interferir con la capacidad de IXP2400 de sostener su performance nominal de procesamiento de paquetes.

La metodología incremental propuesta plantea como primera etapa la verificación del comportamiento del sistema completamente *simulado* en ECD++ bajo un escenario simplificado, con modelos DEVS ejecutando en el XScale Core pero sin interactuar aún con las ME. En la Figura 5.9 (izquierda) se muestra un diagrama conceptual en bloques de los modelos que componen la aplicación de control. Los modelos QoS Actuator y Traffic Sensor en el nivel de baja velocidad se encargan de enviar comandos de control y sensar el estado de TR, respectivamente. Estos bloques se comunican con sus contrapartes en el nivel de alta velocidad (o Packet Processing system), consistiendo en los modelos QoS Shaper y Metering System. Habiendo verificado la funcionalidad básica del QoS Controller en este escenario simplificado utilizando tráfico sintético generado con modelos DEVS, se procede a la etapa de validación del controlador, comunicando los niveles de baja velocidad (modelos DEVS que se conservan intactos) con el hardware real de alta velocidad (MiroEngines). En la Figura 5.9 (derecha) el hardware real de procesamiento de paquetes reemplaza al conjunto de modelos que emulaban sus funciones (Generator, Shaper, Pipeline, Meter, Consumer). Esto es posible gracias a los nuevos modelos DEVS tipo Mapper (trafficShaper

y trafficMeter) que actúan como interfaz con IXP2400 por medio de *Variables IXA*.

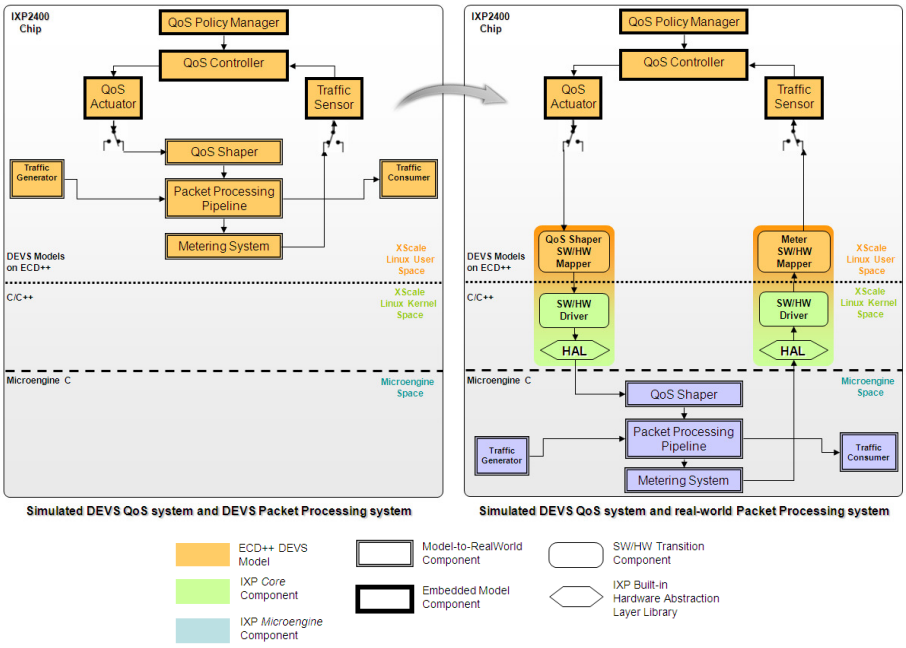


Figura 5.9: Diagrama Conceptual: Transición desde una simulación embebida autónoma (izquierda, fase de verificación) hacia una simulación embebida tipo Hardware–In– The–Loop (derecha, fase de validación).

Se asumirá que existe un algoritmo de bajo nivel ejecutando en las MicroEngines con la capacidad de aplicar técnicas Control de Admisión (AC) de descarte de paquetes para garantizar una longitud media de cola. Dicha longitud es un parámetro de operación que se modifica ante cada *acción de control* proveniente del Control Supervisorio. A su vez, las ME tienen la capacidad de hacer accesible al Control Supervisorio la información actualizada de la cantidad de paquetes transmitidos a cada instante.

El sistema de Calidad de Servicio (QoS) que se desea diseñar con ECD++ debe enviar comandos de control al AC indicando la nueva longitud de cola promedio que se desea obtener. Esta indicación, a su vez, puede variar según la política configurada en el controlador. La Figura 5.10 muestra una representación de este modelo a modo de una máquina de estado. Los 4 estados reflejan la condición del sistema de procesamiento de paquetes, combinando la métrica TR con la cantidad de *tiempo ininterrumpido* T (persistencia temporal) durante el cual se sostiene un nivel dado de TR. En la figura se muestra el diseño visual del sistema de control completo con la herramienta avanzada CD++Builder, quedando ECD++ compilado y listo para interactuar con las ME. Esta fase representa el caso práctico correspondiente a la etapa conceptual de validación a la derecha de la Figura 5.9.

El controlador usa un umbral de tasa de tráfico *rateThreshold* para clasificar la intensidad de tráfico (alta o baja). Inicialmente, si se sensa $TR >$

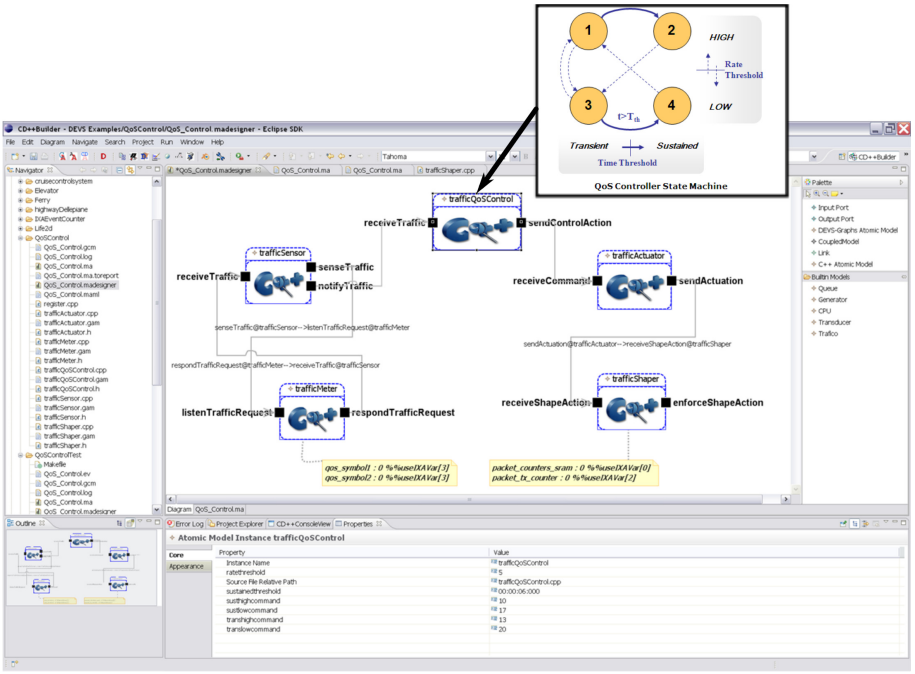


Figura 5.10: Herramienta avanzada de edición de modelos para ECD++. Modelo QoS_Control para IXP2400.

rateThreshold se pasa al estado ① y el tiempo T comienza a incrementarse desde cero. Si en cambio $TR \leq rateThreshold$, el controlador pasa al estado ③. Se define el umbral de tiempo *sustainedThreshold* para distinguir entre 2 posibles valores de persistencia de la tasa TR: Transient y Sustained. Cuando T cruza el umbral *sustainedThreshold*, el modelo evoluciona desde ① a ② o desde ③ a ④ dependiendo si el sistema se encontraba en un nivel de TR alto o bajo, respectivamente. Cuando la tasa de tráfico cruza el umbral *rateThreshold*, se reinicia T a cero y el estado regresa a ① o ③, según corresponda.

Ambos umbrales se definen como parámetros del modelo atómico trafficQoS-Control, como puede observarse en la pestaña Properties al pie de la Figura 5.10, en donde se configuran $sustainedThreshold = 6$ segundos y $rateThreshold = 5$ paquetes/segundo.

En el mismo lugar se observan los parámetros $transHighCommand = 23$, $sustHighCommand = 10$, $transLowCommand = 20$ y $sustLowCommand = 17$. Estos son los comandos que deberán enviarse hacia el AC cada vez que el controlador experimente un cambio en su máquina interna de estados (para los estados ①, ②, ③ y ④ respectivamente). El encargado de enviar efectivamente dichos comandos es el bloque trafficActuator, el cual desconoce la diferencia entre un modelo DEVS estándar y uno tipo Mapper con capacidad de comunicarse con las MicroEngines. Esta función es cumplida por el bloque trafficShaper, el cual declara las *Variables IXA* correspondientes para poder invocar la función SetIXAVar().

Siguiendo los pasos descritos en secciones anteriores del presente capítulo,

se generó el archivo ejecutable para el sistema QoS_Control incorporando las nuevas bibliotecas para dialogar con las MicroEngines, y se lo portó al IXP2400 para realizar ensayos de validación.

El experimento consiste en situar un generador de tráfico en el servidor LinuxHost del laboratorio (IP 192.168.25.3, ver Figura 5.8 generando paquetes hacia un Cliente de Test (192.168.25.5). La ruta implicará que el tráfico pase por la placa RadiSys ENP-2611, y la consecuente actuación del sistema de control diseñado reaccionando a la medición de un tráfico real de paquetes.

El resultado de los mismos se muestra en la Figura 5.11. Allí puede verificarse el nivel de tasa promedio TR medido por trafficSensor/trafficMeter cada 1 segundo (secuencia temporal superior, puerto receiveTraffic). Acorde al comportamiento esperado de la máquina de estados, puede validarse la secuencia de comandos enviados hacia las MicroEngines (secuencia temporal inferior, puerto receiveShapeAction). A su vez, en la secuencia temporal del puerto receiveCommand se denotan los estados que va a asumiendo la máquina de estados de trafficQoSControl.

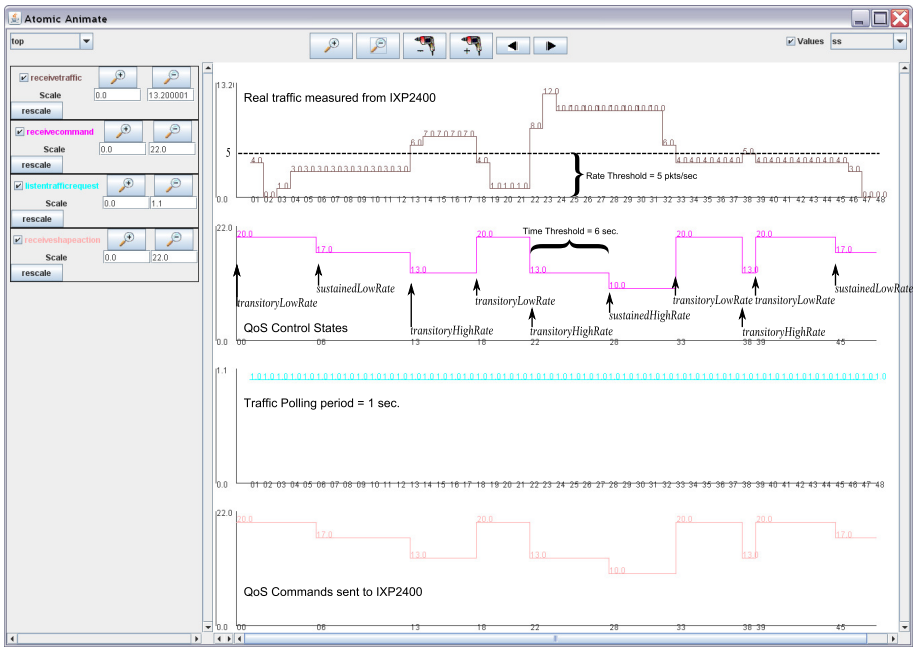


Figura 5.11: Resultados de simulación en tiempo real con tráfico real en modo Hardware In The Loop.

Se observa el comportamiento esperado para el Control Supervisorio, reaccionando con el correcto envío de comandos ante los cruces de umbrales. Se valida así el sistema tanto ante los cambios de nivel de la tasa sensada (que cruzan el umbral de tasa de 5 paquetes/segundo), como ante la permanencia del tráfico en un nivel por más tiempo del establecido por el umbral de permanencia (6 segundos).

5.5. Conclusiones

El presente capítulo introdujo aportes para la implementación de controladores embebidos en tiempo real, y su experimentación sobre plataformas reales. Se aportaron herramientas y metodologías suficientes para obtener un procedimiento intuitivo completamente basado en modelos DEVS, que puedan ser embebidos en un hardware de destino final, ejecutados en tiempo real, e interconectados con unidades externas especializadas. Se garantiza la continuidad de los modelos DEVS que definen a un controlador, al encapsular las funcionalidades específicas de interacción con el hardware externo en modelos atómicos especiales (adaptables para interactuar con interfaces de comunicación propias de cada dispositivo). Esto se hizo posible gracias al desarrollo de nuevas bibliotecas de interfaz para comunicar ECD++ con el procesador híbrido de red IXP2400. Asimismo, mediante nuevas herramientas gráficas avanzadas se logra asistir al proceso de diseño de sistemas embebidos para control de redes. Se diseñó un laboratorio virtualizado y portable para realizar experimentos de modo rápido y simplificado con entornos de configuración compleja, permitiendo trabajar con una placa de red de alta capacidad basada en IXP2400. Los ejemplos prácticos estudiados demuestran que pueden llevarse a cabo con éxito las etapas finales de implementación y validación de controladores de red basados en DEVS operando en tiempo real, eliminando completamente la necesidad de adaptación de lógica ni de estructura al pasar desde las simulaciones autónomas (para propósitos de verificación funcional) hacia las ejecuciones en modalidad Hardware–In–The–Loop (para validación e implementación final). La metodología utilizada garantiza la continuidad de los modelos y promueve el desarrollo de soluciones ingenieriles completamente basadas en modelado y simulación.

Capítulo 6

Aplicación a un Problema de Control de Admisión.

En el presente capítulo se presenta una metodología integral de modelado y simulación para control híbrido de redes de datos. El método es utilizado para el análisis, diseño e implementación de sistemas control Calidad de Servicio (QoS, por *Quality of Service*) en aplicaciones embebidas en Procesadores de Red (Network Processors - NP). Se aplicará teoría de control continuo a estrategias de Control de Admisión (Admission Control - AC) de tráfico de red, es decir a un sistema de eventos discretos. El problema es de naturaleza híbrida, y debe ser tratado formalmente para garantizar la aplicabilidad del control continuo a sistemas de eventos discretos. Mediante el uso del formalismo DEVS en combinación con los métodos numéricos de aproximación de sistemas continuos QSS, se obtienen las siguientes ventajas: un marco formal y unificado para analizar y diseñar con precisión modelos híbridos de AC, y la transición directa desde la simulación a la implementación embebida en procesadores de red.

6.1. Introducción

Los sistemas de red embebidos en procesadores consisten en protocolos de comunicación que se ejecutan en hardware de propósito específico y con recursos limitados. El control de sistemas de Cómputo y Red en Tiempo Real (CRTR) es una nueva disciplina que aplica técnicas clásicas de control a sistemas de cómputo y/o de red, los cuales son considerados como los objetos de control [Hel04]. En estos sistemas, el objetivo de control es mantener las métricas de performance (delay, jitter, throughput, utilización) dentro de rangos especificados por requerimientos de QoS. Las acciones de control son eventos discretos que modifican la asignación de los recursos disponibles (pools, buffers, queues, slots) a las entidades que compiten por ellos (tasks, packets, jobs, requests).

Si bien estos sistemas de control son usualmente diseñados con técnicas ad–hoc, la teoría clásica de control es cada vez más utilizada por su robustez teórica y metodológica [ARH+06]. Así por ejemplo, la estabilidad y respuesta transitoria pueden ser sistemáticamente analizadas, y los controladores pueden diseñarse con criterios óptimos y/o robustos. En la última década, este enfoque ha sido adoptado para diversos sistemas CRTR como routers, clusters de cómputo

intensivo, servidores web, servidores multimedia, etc. [Hel04].

Una dificultad para aplicar teoría de control a estos sistemas es la naturaleza heterogénea de los modelos utilizados para su análisis. Los sistemas CRTR se describen típicamente como Sistemas Dinámicos de Eventos Discretos (DEDS) [CL04] utilizando lenguajes de modelado como Redes de Petri, Autómatas o Máquinas de Estados Finitos. En cambio, la teoría de control clásica se basa en modelos de tiempo continuo (ecuaciones diferenciales) o tiempo discreto (ecuaciones en diferencia). Si bien existen muchas técnicas para el control de sistemas de eventos discretos [CL04], sus objetivos son obtener garantías de temporización y/o seguridad, sin considerar propiedades de estabilidad, respuesta transitoria y robustez ante la dispersión de parámetros.

Para posibilitar la aplicación de técnicas analíticas de control basadas en tiempo a representaciones de sistemas basadas en eventos, los sistemas suelen aproximarse con modelos de tiempo continuo o discreto. Luego, se diseñan y analizan los controladores en el dominio temporal. Sin embargo, estas aproximaciones conllevan sobresimplificaciones y errores, por lo que los controladores deben ser posteriormente verificados (y eventualmente rediseñados) utilizado modelos más precisos, es decir, modelos de eventos discretos. Si bien existen herramientas avanzadas [IH08] para esta fase, su integración con los modelos de control continuo es muy compleja, forzando adaptaciones ad–hoc de software que fue diseñado sin considerar este objetivo, y sin contar con el respaldo de un formalismo que garantice la correctitud de la tarea.

El proceso de diseño requiere modelos de distinta naturaleza, descritos con diversos lenguajes, elaborados por expertos de diversas áreas. Como resultado, se debe transicionar entre distintas técnicas y herramientas de modelado y simulación, causando dificultades prácticas. Incluso la implementación de los algoritmos de control en modo embebido en procesadores plantea desafíos particulares, ya que el código debe ser adaptado a la plataforma de destino.

En el presente capítulo se propone una metodología integral para resolver estos desafíos, aprovechando las capacidaes de DEVS para describir y simular cualquier sistema híbrido.

Se ilustrará la metodología aplicándola al análisis, diseño e implementación de un sistema de control de QoS embebido en un NP Intel IXP2400 [Int04].

Se analizará un caso de estudio en estrategias de AC de tráfico de red, basadas en teoría de control no lineal. Se obtendrá un algoritmo de AC como una secuencia de tareas de modelado y simulación. Sintéticamente, primero se modela la red con una aproximación de tiempo continuo (diseñando una ley de control en tiempo continuo). Luego se halla una aproximación de tiempo discreto para la ley de control (verificando el controlador discreto con un sistema de red a eventos discretos). Por último se descarga el modelo del controlador al NP, en el cual reside un el simulador DEVS embebido ECD++ con capacidad de ejecución en tiempo real.

6.2. Antecedentes

Las redes de comunicación modernas se plantean como sistemas estructurados en capas. En ellas, los servicios ofrecidos por una capa inferior hacia su capa superior, pueden pensarse como provistos por un *Service Control Node* genérico, compuesto por una *lógica de servicio* y una *lógica de control*. La lógica de

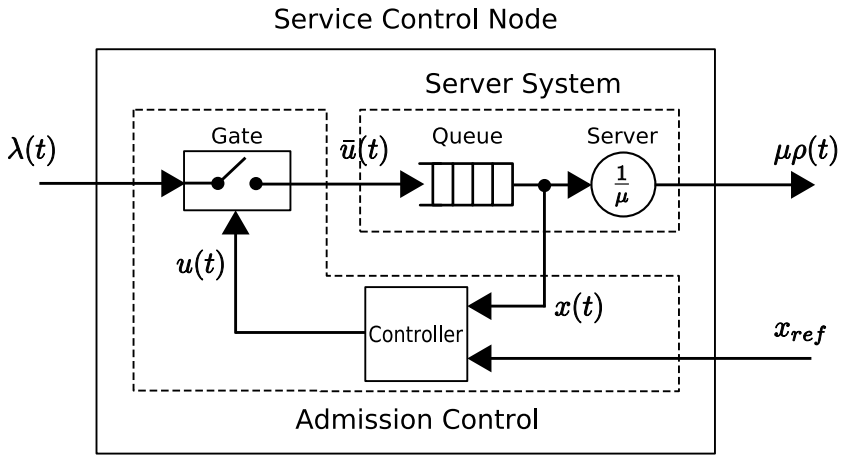


Figura 6.1: Caso de estudio: Service Control Node.

servicio es provista por servidores (*servers*) que procesan paquetes de red a una tasa de servicio (*service rate*) dada. Los paquetes que no pueden ser procesados son ubicados en colas (*queues*). La lógica de control intenta hacer cumplir los requerimientos de QoS bajo condiciones de congestión (el estado de la red en el cual los objetivos de QoS no puede ser totalmente satisfechos). Esta situación es manejada por mecanismos de control de congestión [KK00], siendo los más conocidos el control de tasa para redes ATM (capa de enlace) y el control de ventana deslizante para redes TCP (capa de transporte).

El AC es una lógica de control particular basada en el rechazo selectivo de paquetes antes de ser encolados, mientras que aquellos paquetes aceptados tienen garantizado el acceso al servidor. La teoría de control es cada vez más utilizada para describir, analizar y diseñar algoritmos robustos de AC. El control no lineal ha sido aplicado al diseño de controladores de admisión en [KRW03]. Allí los autores recurren a distintas técnicas y herramientas de modelado y simulación para tratar con la naturaleza híbrida del sistema. En el presente capítulo se mostrará como obtener los mismos resultados utilizando una metodología integrada, con el potencial de mejorar la confiabilidad, el reuso y la distribución de los modelos, y de facilitar la transición hacia el hardware destino.

6.3. Caso de Estudio

Se presentará un caso de estudio que consiste en obtener un AC basado en teoría de control. Se adoptará el ejemplo mostrado en la Fig. 6.1 para un **Service Control Node** presentado en [KRW03].

El bloque **Server System** es la planta a ser controlada. Se compone de un bloque **Queue** y un bloque **Server**. El requerimiento de QoS es mantener la longitud de cola en un valor de referencia (es decir, el *set point* x_{ref} para la planta), siendo éste el objetivo de control. La señal de entrada a la planta es la tasa de admisión (*admittance rate*) $\bar{u}$ de paquetes a la cola, que se asume sigue un proceso estocástico Markoviano. El bloque **Queue** se asume con capacidad infinita, y el número instantáneo de paquetes en la cola es la *señal de salida* x de

la planta. Los tiempos de servicio en el server obedecen a cierta distribución de probabilidades con un valor medio de $1/\mu$ segundos. La naturaleza estocástica de la tasa de admisión y de los tiempos de servicio representa el ruido que debe ser compensado por el bloque **Admission Control**. El bloque **Controller** mantendrá el número de paquetes encolados x en el valor deseado x_{ref}. La acción de control consiste en abrir y cerrar una compuerta a la entrada del **Server System** para limitar el tráfico de ingreso (*arrival rate* λ). Esto se logra con el bloque **Gate**, el actuador del control. Dependiendo de la *señal de control* u, la compuerta rechaza paquetes selectivamente del tráfico λ para controlar el admittance rate $\bar{u}$.

Para aplicar teoría de control al **Server System**, se puede obtener su modelo en ecuaciones diferenciales [KRW03] según:

$$\frac{dx}{dt} = \bar{u}(t) - \mu G(x(t)) \tag{6.1}$$

donde $\bar{u}(t)$ es el admittance rate a la cola, $1/\mu$ es el tiempo medio de servicio y $G(x)$ es una función no lineal:

$$G(x(t)) = \frac{x(t) + 1 - \sqrt{x^2(t) + 2C^2x(t) + 1}}{1 - C^2} \tag{6.2}$$

C es el coeficiente de varianza de los tiempos de servicio. Esta aproximación reproduce con precisión el estado estable del número promedio de paquetes y la utilización del server; y captura parte de las propiedades estocásticas bajo condiciones de tráfico no estacionario. La acción de la compuerta puede describirse según:

$$\bar{u}(t) = \begin{cases} \lambda(t) & \text{si } u(t) > \lambda(t) \\ \text{máx}(0, u(t)) & \text{en cualquier otro caso.} \end{cases} \tag{6.3}$$

Los autores propusieron un controlador PI para el bloque **Control Node** (ver Fig.6.2) con la siguiente ley:

$$u(t) = Ke(t) + \frac{K}{T_i} \int e(t)dt \tag{6.4}$$

donde $e(t) = x_{ref} - x(t)$. Los parámetros del sistema para este ejemplo son: $\lambda = 20s^{-1}$, $x_{ref} = 10$, $\mu = 5s^{-1}$, $\mu_1 = 2s^{-1}$, $\mu_2 = 60s^{-1}$, $\alpha_1 = 0{,}38$ y $C^2 = 3{,}7$. Los tiempos de servicio en el server están distribuidos hiperexponencialmente, combinando 2 distribuciones exponenciales con tiempos de servicio promedio μ_1 y μ_2 respectivamente, con un desvío de $\alpha_1 = 38\,\%$ a favor de μ_1. Esto resulta en una tasa media de servicio de $\mu = 5\ sec^{-1}$ con un coeficiente cuadrático de varianza de $C^2 = 3{,}7$. El diseño se completa encontrando valores convenientes para K y T_i. En este caso, siguiendo un procedimiento de linealización y ubicación de polos, estos resultan $K = T_i = 2{,}4$. El sistema así parametrizado será referido como el *Controlled Packet Processing System* (CPPS).

6.4. Metodología basada en DEVS

En esta sección se muestra el uso de la metodología propuesta basada en DEVS para asistir el proceso de análisis, diseño, verificación, implementación

y validación del controlador PI especificado para el CPPS. Se reproducirán los principales resultados del trabajo original de [KRW03] para validar nuestro enfoque.

Las etapas de análisis, diseño y verificación son asistidas por la herramienta *PowerDEVS* aprovechando su capacidad para modelado y simulación de sistemas híbridos. Las fases de implementación y validación en la plataforma de hardware destino son asistidas por la herramienta Simulador CD++ Embebido (ECD++, por *Embedded CD++ Simulator*), una extensión de la herramienta de modelado y simulación de propósito genérico CD++. ECD++ tiene capacidad para ejecutar en tiempo real embebido en procesadores de propósito específico [Wai09]. Gracias a su diseño basado en DEVS, ambas herramientas pueden simular los modelos obtenidos, haciendo que la transición sea directa (a nivel de su especificación formal).

6.4.1. Control Continuo. Planta Continua.

El primer objetivo del proceso es verificar los parámetros de control del CPPS vía simulación. En este punto, se cuenta con una aproximación no lineal en tiempo continuo del Server System, y una especificación en tiempo continuo para Controller.

En la Fig. 6.2 se muestra el modelo de tiempo continuo del CPPS construido en PowerDEVS. Cada bloque en el modelo representa un modelo DEVS *atómico* o *acoplado*. Los detalles internos de los modelos acoplados **Server System** y **PI-Controller** se muestran en la misma figura. El modelo atómico **Non-Linear Function** (un componente de **Server System**) implementa la expresión no lineal del lado derecho de la igualdad en Ec.(6.1). Los modelos **QSS Integrator** implementan los métodos QSS, en este caso con precisión de orden 3 (QSS3). Los modelos utilizados son parte de la biblioteca standard de PowerDEVS.

Se ejecuta la simulación de este sistema por 30 segundos. Los resultados se muestran en la Fig. 6.3 adonde puede verse una respuesta estable y levemente subamortiguada de la señal Queue Length. Estos resultados, obtenidos en un entorno de eventos discretos, coinciden cualitativamente con los resultados de [KRW03] (obtenidos con aproximaciones de tiempo discreto) para los mismos parámetros y rangos de señal.

6.4.2. Control Discreto. Planta Continua.

El segundo objetivo del proceso de diseño es obtener un controlador que pueda ser implementado con un algoritmo ejecutable en un sistema digital. Con este fin, se traduce el controlador a una ley en tiempo discreto, pero manteniendo (por ahora) la versión a tiempo continuo de la planta. Aplicando el método de Euler a (6.4) se obtiene la siguiente expresión a tiempo discreto para el controlador PI:

$$z_{k+1} = z_k + e_k h \quad u_k = K(e_k + \frac{z_k}{T_i}) \tag{6.5}$$

donde $k \in Z$ es el subíndice de tiempo discreto y $h \in \Re^+$ es el paso de tiempo, con $z_k = z(t = kh)$. Luego, se reemplaza la versión original de **PI-controller** por su equivalente discretizado, obteniendo la Ec.(6.5). El diagrama de bloques del controlador discreto se muestra en la Fig. 6.4. Las versiones discretas de las

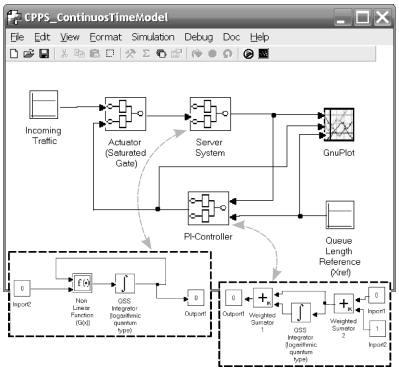


Figura 6.2: Modelo continuo del Controlled Packet Processing System en PowerDEVS.

variables x_k y x_{ref_k} se obtienen con bloques **Sample and Hold** parametrizados con un período de muestreo de $h = 0{,}5$ segundos.

Se ejecuta la simulación del nuevo sistema con el control discreto durante 30 segundos. Los resultados se muestran en la Fig. 6.5. El análisis de este resultado muestra que se reproducen satisfactoriamente las propiedades del sistema original de tiempo continuo. Luego, se adopta a este diseño para el control discreto como una solución viable.

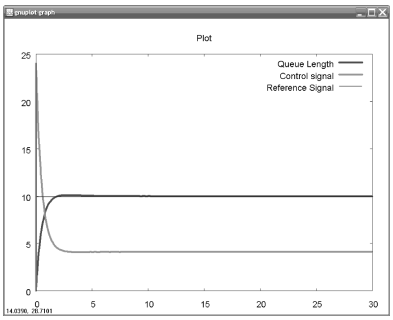


Figura 6.3: Simulación del Controlled Packet Processing System a tiempo continuo.

6.4.3. Control Discreto. Planta Discreta.

Siguiendo un ciclo de diseño típico de modelado y simulación, se debe verificar el diseño del control discreto funcionando con una representación más precisa del sistema real, tratando de evitar –tanto como sea posible– realizar aproximaciones. Este es el tercer objetivo de la metodología propuesta. El Server System es en rigor un sistema de colas tipo M/G/1 (un sistema estocástico de eventos discretos). Luego, puede ser directamente representado y simulado

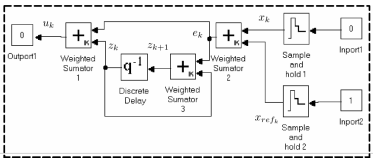


Figura 6.4: **PI-Controller** discreto en PowerDEVS.

con el formalismo DEVS. Una de las ventajas de esta metodología es que la actividad de verificación contra el sistema original involucra un simple reemplazo de modelos (sin cambio de formalismo ni de herramienta). Luego, se reemplazan los componentes continuos por sus equivalentes a eventos discretos; obteniendo el modelo híbrido de la Fig. 6.6.

Por el contrario, en una metodología tradicional, este paso involucraría algún tipo de *implementación específica* del modelo híbrido (es decir, del sistema de colas M/G/1 y del algoritmo de control). El código obtenido es usualmente de

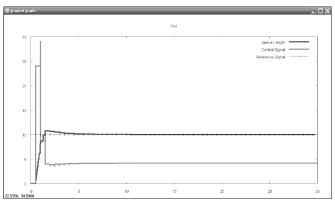


Figura 6.5: Simulación del Controlled Packet Processing System a tiempo discreto.

propósito especial y dependiente del lenguaje utilizado, disminuyendo la reusabilidad y potenciando la introducción de errores. En el caso de [KRW03] los autores desarrollaron un código específico en el lenguaje C.

El modelo Hybrid CPPS de la Fig. 6.6 implementa los siguientes modelos atómicos DEVS para representar el sistema de red: **Packet Generator**, **Admission Gate**, **Queue** y **Server**. Estos modelos manipulan entidades discretas individualmente (los paquetes de red) y lo hacen a tiempo continuo. El com-

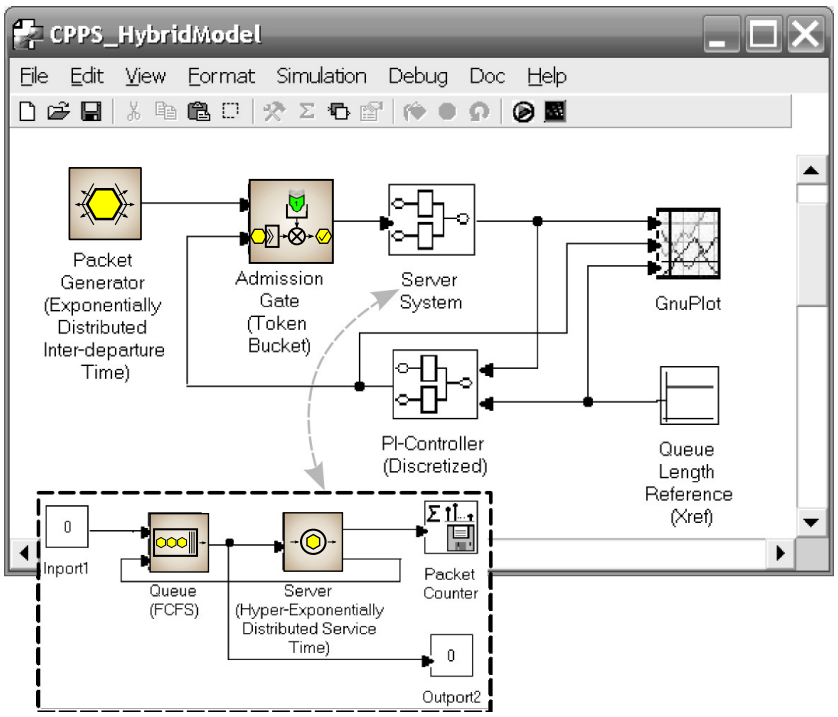


Figura 6.6: Modelo híbrido del Controlled Packet Processing System en PowerDEVS.

portamiento estocástico del tráfico de red se considera explícitamente en los tiempos de inter–generación de paquetes y de procesamiento de los mismos. De este modo, no existe ninguna aproximación para la dinámica estocástica del sistema.

El encolamiento de los paquetes se realiza según la política *First In-First Out* (FIFO) en el bloque **Queue**. Un algoritmo denominado *token bucket* es implementado en el bloque **Admission Gate** con el fin de generar "tickets" internos a una tasa dada. Esta tasa es determinada por la señal de control u. Cuando los paquetes arriban a la compuerta, serán autorizados a continuar en el caso que existan tickets disponibles (cada paquete autorizado consume un ticket). Los paquetes que no encuentren tickets disponibles son rechazados. Este algoritmo es exactamente igual al que debería ser implementado en un código embebido en un procesador de red real.

Se realizaron 1000 simulaciones de 30 segundos cada una y se obtuvo su promedio. Los resultados se muestran en la Fig. 6.7. Se observa la curva promedio del ensamble de señales Queue Length y también la curva de una realización particular representativa del conjunto. Ambas curvas fueron obtenidas con un muestreo de señal cada 1 segundo. La curva punteada denota una regulación satisfactoria del promedio de Queue Length entorno al valor de referencia deseado (10 paquetes). Una comparación cualitativa de estos resultados con los obtenidos en [KRW03] para los experimentos de eventos discretos (bajo las mismas condiciones) muestra una coincidencia muy satisfactoria.

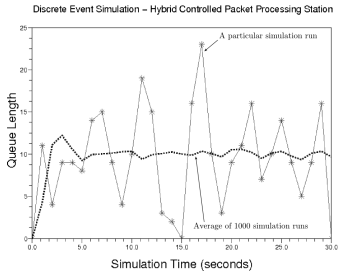


Figura 6.7: Simulación del Controlled Packet Processing System a modo híbrido.

6.4.4. Implementación Embebida en Red

La metodología presentada permite implementar el controlador basado en DEVS desarrollado en las secciones anteriores, embebido en un NP para aplicaciones de red de alta performance. Se busca evitar en la mayor medida posible recodificar la solución diseñada, portando el diseño del bloque **Controller** directamente desde PowerDEVS a un motor de simulación capaz de ejecutar modelos DEVS embebido en el NP.

Con este fin, se utilizaron los nuevos métodos y herramientas descritos en el

Capítulo 5 para portar el modelo **Controller** al entorno ECD++ ejecutando en tiempo real y embebido en el NP IXP2400.

Se utilizaron técnicas de traducción semiautomáticas (de PowerDEVS a ECD++) para obtener los modelos atómicos equivalentes, y se utilizó la nueva interfaz gráfica de modelado de ECD++ para definir el acoplamiento de dichos modelos, con resultados satisfactorios. Concretamente se tradujeron los bloques atómicos: suma, delay, generador de señal constante, y sample and hold.

Si bien este procedimiento aún no satisface el caso ideal de obtener una traducción completamente automática (tanto para modelos atómicos como acoplados) y libre de fallas, las herramientas utilizadas disminuyen notablemente la probabilidad de cometer errores en la transición de los modelos hacia su implementación en hardware.

Cualquiera sea el procedimiento de traducción (más o menos automatizado) la estrategia basada en DEVS ofrece una solidez formal que deja perfectamente claro cuales son las reglas de acoplamiento y de manejo del tiempo, que deberán respetarse para garantizar la equivalencia estructural entre un modelo de partida y su versión traducida. En cuanto a la parte comportamental, si dos simuladores implementan sus modelos con el mismo lenguaje de programación (este es el caso entre ECD++ y PowerDEVS, usando C++), se anula completamente la posibilidad de introducir errores –como mínimo– en aquellas porciones de código que implementan la lógica interna de cada función DEVS. Es decir, es posible reutilizar bloques completos de código sin necesidad de reescribirlos.

Evidentemente quedan muchas tareas por realizar en este aspecto. Se tornan muy importantes aquí los esfuerzos en curso de la comunidad DEVS para acordar una estandarización de especificación declarativa de modelos, pasible de ser procesada por algoritmos de traducción, o incluso de generación.

6.5. Conclusiones

Se mostró una metodología integral basada en DEVS para aplicar teoría de control a sistemas de admisión de paquetes en redes de datos utilizando modelos híbridos estocásticos. Se logró eludir los problemas que se presentan al tratar con sistemas híbridos debido a la necesidad de recurrir a diversos paradigmas y herramientas de modelado y simulación. Se combinaron sistemas continuos con sistemas de eventos discretos en un marco unificado apto para integrar disciplinas heterogéneas en el área de control de sistemas de cómputo y/o red en tiempo real. Se logró la transición directa entre herramientas basadas en DEVS, facilitando la implementación de modelos embebidos en un procesador de red Intel IXP2400. Esto reduce riesgos y esfuerzos de recodificación.

Capítulo 7

Aplicación a un Problema de Control de Congestión

El presente capítulo muestra la utilización de PowerDEVS con los nuevos métodos DQSS para modelar, mediante un procedimiento sencillo e intuitivo, una representación continua del Control de Congestión tipo TCP/AQM. Si bien esta representación limita el estudio al comportamiento promediado de las variables de interés, permite comprender en tiempos de simulación muy convenientes características importantes del sistema como ser estabilidad, convergencia, robustez ante variación de parámetros, etc. Sin embargo, el precio que se paga es la pérdida de información detallada a nivel paquetes individuales, como por ejemplo la distribución probabilística de los retardos punta a punta. Luego, usando la misma herramienta, se desarrollan modelos de eventos discretos para simular de manera exacta el comportamiento de la red bajo el esquema TCP/AQM, usando esta vez primeros principios de los protocolos y medios físicos. Para este objetivo se utilizan las nuevas bibliotecas tanto para representar estructuras de paquetes de propósito genérico, como para la construcción de sistemas estocásticos STDEVS mediante generadores de muestras aleatorias para aproximar distribuciones de probabilidad. Los resultados obtenidos en este segundo paso permiten verificar la aproximación fluida realizada inicialmente, como así también estudiar detalladamente propiedades discretas. Por último, se implementa una estrategia de modelado de flujos híbridos que logra combinar la ventaja de la eficiencia computacional de una simulación fluída con la ventaja del detalle granular provisto por una simulación discreta. Las propiedades formales de DEVS, STDEVS, QSS y DQSS permiten obtener flujos híbridos bajo un formalismo matemático unificado y en una misma herramienta.

7.1. Introducción y Antecedentes

En la última década se han propuesto diversos controles de congestión para TCP/AQM [Bar02] modelándolo como un sistema de control descentralizado, distribuido y con retardos variables. La importancia de contar con modelos precisos radica en la gran dificultad que presenta el ajuste simultáneo y eficiente de los numerosos parámetros de operación de TCP/AQM.

Si bien las técnicas originales de TCP/AQM fueron muy exitosas en sus

orígenes para redes de capacidad moderada, comenzaron a demostrar limitaciones en los nuevos escenarios de redes con alta capacidad y altas latencias, es decir con alto Producto Retardo–Ancho de Banda (BDP, por *Bandwidth–Delay Product*) [LBS08].

Recordando que el principio de regulación de CA/TCP utiliza un sistema de *ventana deslizante* [CHdV03] que limita el envío de paquetes a la red sólo una cantidad W por ciclo de transmisión, siendo el objetivo de dicha regulación mantener el vínculo lo *más ocupado posible*, pero sin saturarlo. En la condición ideal, se verifica $W = BDP$; es decir, la ventana imita (y *limita*) la cantidad de paquetes "en vuelo" que puede alojar el vínculo (dada su condición de velocidad y retardo, el vínculo es visto como un *elemento almacenador* de paquetes).

Entre otros problemas, el crecimiento y decremento de la ventana de ajuste W presenta una dinámica muy lenta de recuperación ante una detección simple de congestión (la pérdida de un único paquete). Esto produce una subutilización del enlace hasta que W retoma su valor óptimo, siendo justamente la subutilización el fenómeno que busca evitarse con CA/TCP. Además, para redes de alto Producto Retardo–Ancho de Banda (BDP) se incrementa la sensibilidad frente al ajuste de parámetros, una tarea dificultosa, usualmente basada en heurísticas e iteraciones de prueba y error.

La comunidad científica propuso una gama de mejoras de diseño para solucionar estos y otros problemas recurriendo a técnicas analíticas y empíricas. Todas ellas enfrentan el desafío de la credibilidad, ya que se postulan para reemplazar una tecnología crítica que posee actualmente miles de millones de usuarios activos, y millones de enrutadores en servicio. En este contexto, un proceso integral de modelado, simulación, verificación y análisis de sistemas híbridos juega un papel central en el diseño de futuros protocolos de redes de alta capacidad [FK03].

Para elevar la credibilidad de dicho proceso, es vital descansar sobre un formalismo matemáticamente robusto, y con un marco de modelado y simulación que evite tener que cambiar de formalismo al cambiar de paradigma de representación. Evidentemente, DEVS es un candidato natural para cubrir estos objetivos. Se verá a continuación de qué forma DEVS provee una solución unificada para el modelado híbrido de TCP/AQM usando las nuevas herramientas formales STDEVS y DQSS, entre otras.

En principio, el paradigma de eventos discretos es la forma de representación natural para una red y sus algoritmos de control. Pero se presentan dificultades al aplicar las técnicas estándar de simulación de eventos discretos en redes de alta capacidad: independientemente de la eficiencia del simulador que se elija o del grado de optimización algorítmica provista a los modelos, existe una limitación intrínseca de escalabilidad para efectuar simulaciones de centenas de flujos de paquetes a altas tasas de transmisión.

Es aquí donde cobra relevancia el modelado continuo, utilizando aproximaciones fluidas de los flujos de paquetes y de la dinámica de sus algoritmos de control de congestión. El paradigma continuo permite estudiar problemas de gran magnitud en tiempos de simulación razonables y con una capacidad de cómputo modesta, agilizando el proceso de investigación.

Por ello, en la literatura se ha propuesto una variedad de modelos basados en ecuaciones diferenciales no lineales para estudiar la dinámica de TCP [PFTK98, Kel01, GM02, Mas05]. Estas ecuaciones se basan en aproximaciones continuas de magnitudes como tasas, probabilidad de pérdida de paquetes, incertidumbre

del tiempo de retardo entre nodos y de las particularidades estocásticas del tráfico (no estacionariedad, ráfagas, autosimilaridad, etc. [PW00]). En base a estos modelos se desarrollan algoritmos avanzados [HMTG01] para controlar eficientemente longitudes de colas, tasas de transmisión, retardos máximos, etc.

Una característica notable de estas aproximaciones fluidas para controles distribuidos en red es que producen sistemas de Ecuaciones Diferenciales con Retardo (DDE), siendo los retardos a su vez, variables en el tiempo. Estos retardos juegan un rol decisivo en la estabilidad y el desempeño de los controles, por lo cual su simplificación como retardos constantes –o directamente su aproximación a cero– no son alternativas deseables. Existen muy pocas herramientas de simulación para resolver DDE [BTE09] las cuales no proveen una interfaz de especificación intuitiva ni tienen posibilidad de combinar sistemas continuos con sistemas de eventos discretos (en general) ni con sistemas de redes de datos (en particular). Se verá más adelante que esta limitación se soluciona completamente y de modo sencillo recurriendo al nuevo método DQSS presentado en esta Tesis.

7.2. Caso de Estudio

El presente caso de estudio retoma aquel postulado como escenario motivador en la Sección 2.5.1. Se refiere a dicha sección para una discusión más extensa –aunque menos específica– acerca de TCP/AQM.

En esta sección se describirán en detalle los mecanismos subyacentes que definen el Control Distribuido de Congestión (TCP/AQM), con el objetivo de modelarlo y simularlo con técnicas continuas y discretas. Dichos mecanismos son Incremento Aditivo, Decremento Multiplicativo (AIMD, por *Additive Increase Multiplicative Decrease*) (para Control de Flujo en un nodo TCP) y RED (para Control de Admisión en un enrutador).

Se adoptarán una serie de simplificaciones a *nivel lógico* (protocolo y algoritmos de control) sobre los sofisticados mecanismos de TCP. Esto es necesario debido a que una descripción completamente minuciosa no aportaría ningún enriquecimiento adicional a la comprensión de las herramientas desarrolladas ni del sistema analizado. Es decir, se respetará el nivel de detalle necesario para capturar la esencia de la dinámica de TCP/AQM, evitando desvirtuarla. Por ejemplo, una simplificación consistirá en adoptar como unidad mínima de manipulación a un *paquete* de red, cuando en realidad muchas funcionalidades de TCP operan a nivel de bits o bytes.

Sin embargo, a *nivel físico*, los retardos impuestos tanto por el ancho de banda de los canales como por la velocidad de procesamiento de los nodos mantendrán la debida dependencia con el tamaño en bits de cada paquete.

7.2.1. Control de Congestión por Ventana Deslizante. Mecanismo AIMD.

Las versiones estándar de TCP que se encuentran masivamente implementadas son TCP–Reno [PFTK00] y TCP–NewReno [FH99]. Ellas implementan una clase de control de congestión que, entre otros mecanismos, se caracteriza por la dinámica de su fase de *prevención de congestión*. Esta fase es dominada

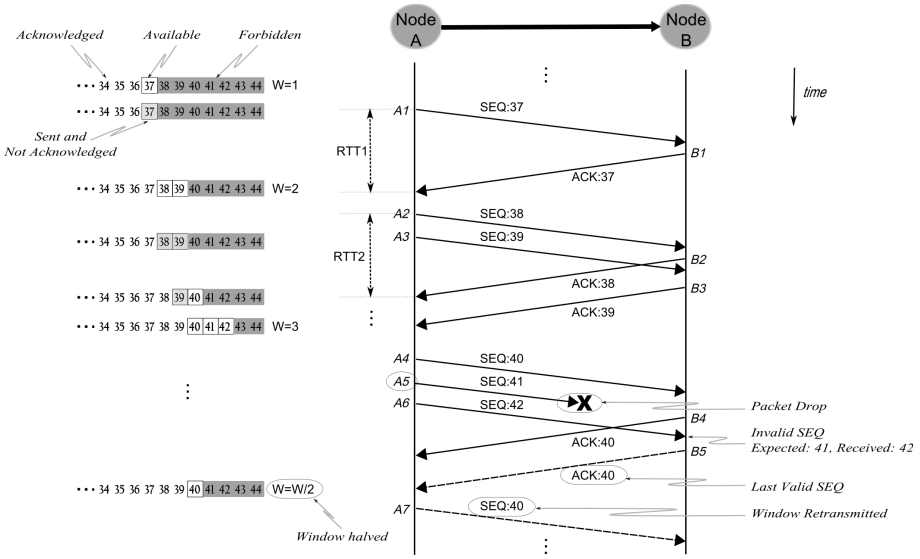


Figura 7.1: Control de Congestión por Ventana Deslizante.

por el algoritmo conocido como Incremento Aditivo, Decremento Multiplicativo (AIMD) [YL00] por medio del cual se ajusta el tamaño W de la ventana de congestión en el extremo emisor de la siguiente manera: W se incrementa en $W = 1/W$ ante la llegada de cada paquete de reconocimiento (ACK) correcto (Additive Increase) y se disminuye en $W = W/2$ inmediatamente que se detecta la llegada de un ACK no esperado (Multiplicative Decrease). Si no llegase ningún ACK luego de tiempo máximo RTO (RTT Time Out) suele forzarse $W = 1$ (sin embargo, en este trabajo se trata a un RTO como un ACK no válido).

Cada flujo TCP mantiene un máximo de W paquetes inyectados en la red sin ser reconocidos con sus correspondientes ACK, incrementando la ventana W en 1 paquete por cada Tiempo de Ida y Vuelta (RTT) en caso de que todos los ACK *en ese lapso* hayan llegado exitosamente. Esta condición equivale a considerar que el estado actual de la red es apto para ser llenada con W paquetes sin ocasionar problemas, y que puede intentarse un próximo ciclo con $W = W + 1$, para probar suerte. Es decir, bajo AIMD el emisor intenta cautelosamente enviar ráfagas cada vez más grandes de paquetes con la esperanza de encontrar ancho de banda disponible y no provocar congestión. En caso de experimentar una pérdida de paquete, se infiere una situación de congestión y se retrotrae el ritmo de envío limitando el tamaño de (W) a la mitad del último utilizado. De este modo, en cada instante, una conexión TCP limita su tasa de transmisión de paquetes a $\lambda_{max}^{TCP}(t) = \frac{W(t)}{RTT(t)}$. Se trata de un esquema bastante conservador, que se actualiza a un ritmo dominado por el retardo RTT, y que limita su tasa a la soportada por el elemento más lento de la red (aquel enrutador que sea el cuello de botella de una ruta).

Para ilustrar el mecanismo, en la Figura 7.1 se muestra una secuencia de intercambio de paquetes entre un Emisor A y un Receptor B. Se observa un

sistema habiendo enviado exitosamente paquetes hasta la secuencia SEQ:36, y se supone que por algún motivo la ventana se encuentra en W=1. La ventana se indica en el margen izquierdo junto a la secuencia de identificadores de paquetes (ID). En fondo blanco se muestran los ID ya enviados y con su ACK correcto recibido. En fondo gris oscuro se muestran los IDs que no pueden ser utilizados por más que exista información disponible para enviar. En recuadro de línea se muestra la ventana W. Un ID de la ventana con fondo blanco es un ID disponible que aún no se utilizó, mientras que un ID de ventana con fondo gris es uno que se utilizó pero aún se está aguardando su ACK (son los paquetes "en vuelo"). Como se observa, la ventana se va deslizando hacia IDs superiores a medida que llegan paquetes ACK desde B.

En una situación normal, la llegada a B de un número de secuencia esperado produce la devolución del mismo número en forma de ACK hacia el emisor [1]. Esta dinámica se observa desde el envío del paquete A1 hasta la recepción del paquete A4. Si se obtienen todos los ACK correctos para una ventana completa de transmisión, el emisor aumenta W en 1. Esto se observa en la llegada de los paquetes B1 y B3.

Sin embargo, cuando el receptor recibe un k–ésimo paquete con secuencia SEQ_k distinta al esperado inmediato siguiente $SEQ_{k-1}+1$, devuelve al emisor el último ACK válido, duplicándolo. Un ACK no esperado por el emisor es en general un ACK repetido o "duplicado" (Duplicated ACK - DACK). Este es el caso de B5. Debido a la pérdida del paquete con SEQ:41, el paquete B5 se encarga de señalizar la situación al emisor, el cual reduce su ventana con $W = W/2$ y comienza a retransmitir la ventana desde su primer paquete (en este caso, desde SEQ:40)[2].

Sucesivas repeticiones de pérdida de paquetes produce múltiples DACKs. En algunas implementaciones la indicación de congestión no es por la llegada del primer DACK (como se mostró en la Figura 7.1) sino por N (múltiples) DACKs (N-DACK). En los modelos de este trabajo se adoptará la práctica generalizada 3-DACK.

En la Figura 7.2 se muestra una porción de evolución típica de $W(t)$, en donde se detectan 3 pérdidas de paquetes. La pendiente de crecimiento durante las fases de crecimiento aditivo (lineal) queda determinada por RTT.

En la siguiente sección se describe el mecanismo de control de admisión que sucede en los enrutadores, y que son los responsables principales de que ocurran los eventos de pérdida de paquete.

7.2.2. Administración Activa de Colas (AQM). Mecanismo Random Eearly Detection.

El mecanismo de AQM más adoptado para operar con TCP/AQM es Detección Temprana Aleatoria (RED, por *Random Early Detection*) [FJ02] por medio del cual cada enrutador monitorea su cola local, y cuando detecta que la congestión es inminente envía implícitamente una señal al emisor descartan-

[1] En rigor, un código de ACK transporta el *próximo número de SEQ que el emisor está autorizado a enviar*, y de hecho dicho número se refiere a un contador de bytes, en lugar de paquetes. Sin embargo a los efectos del análisis realizado aquí, es suficiente la descripción propuesta.

[2] En rigor, existen métodos como SACK (Selective Acknowledge) y/o Fast Retransmit para evitar reenvíos de ventanas completas. En el presente análisis, no se tendrán en cuenta.

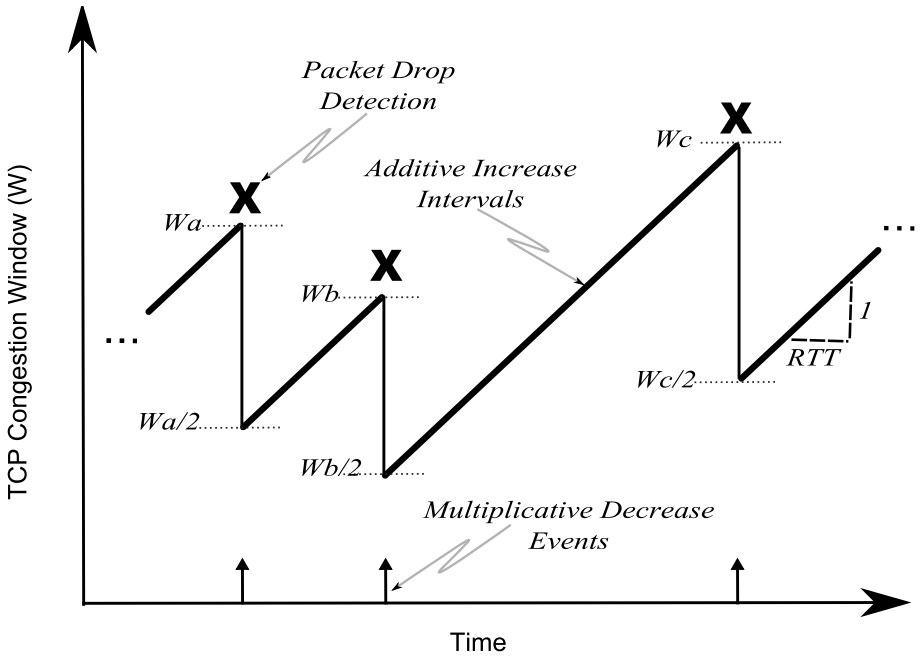


Figura 7.2: Evolución temporal de la Ventana de Congestión W.

do un paquete, situación que provocará la generación de DACKs por parte de AIMD en el nodo receptor. RED no espera a que el enrutador sature su cola, sino que comienza a descartar paquetes de manera temprana y esporádica, con la intención que AIMD el nodo transmisor reduzca preventivamente su ventana W. RED calcula una longitud de cola promedio x aplicando un promedio móvil ponderado (filtro pasabajos): $x = (1-\alpha)x + \alpha x_{inst}$, en donde x_{inst} es la última actualización de la longitud instantánea de la cola, y $0 < \alpha < 1$ es la constante de tiempo del filtro.

El uso de x en lugar de x_{inst} captura más eficientemente el concepto de congestión. Debido a la característica de ráfagas del tráfico de internet, las colas se llenan y vacían muy rápidamente. Conociendo que la dinámica de reacción más veloz a la que puede aspirar AIMD es de 1 RTT, no tiene sentido enviar notificaciones de congestión demasiado reactivas, ya que en general obedecerían a una ráfaga y no a una medición representativa de llenado. El filtro pasabajos intenta detectar *condiciones persistentes* de congestión.

RED posee dos umbrales de longitud promedio de cola, uno mínimo t^{min} y uno máximo t^{max} a los cuales les asigna las probabilidades de descarte de paquetes $p = 0$ y $p = p^{max}$ respectivamente. La función $p(x)$ se muestra en la Figura 7.3.

Ante cada modificación de la longitud instantánea x_{inst} de la cola (ya sea por partida de una paquete o por la llegada de un nuevo paquete), se aplica el siguiente algoritmo:

- **Recalcular** $x = (1-\alpha).x + \alpha.x_{inst}$
- Si $x \leq t^{min}$

Encolar el paquete.

Es decir, no se toma ninguna acción de señalización.

- Si $t^{max} < x < t^{min}$

  Recalcular $p(x) = p^{max}.(x - t^{min})/(t^{max} - t^{min})$

  Descartar el paquete con probabilidad p.

  Es decir, se comienza a señalizar una potencial congestión, proporcionalmente a x.

- Si $x > t^{max}$

  Descartar el paquete.

  Es decir, la estrategia de señalización preventiva no dió resultado y se necesita una medida drástica. (señalizar sí o sí anta cada nuevo paquete)

Si todo marcha bien, RED elimina un pequeño porcentaje de paquetes cuando x excede levemente a t^{min}, y al cabo de 1 RTT el AIMD emisor reduce su tasa de envío, lo cual hace disminuir a x. De este modo, la congestión se elimina, la longitud de cola se mantiene pequeña, y la tasa de tráfico efectivo se mantiene alta, utilizando eficientemente los recursos disponibles. Cabe notar que la longitud instantánea x_{inst} puede ser mucho mayor a x y a t^{max}, permitiendo así alojar ráfagas momentáneas sin alterar el comportamiento promedio del control. Las guías de diseño suelen sugerir adoptar $t^{max} = 2.t^{min}$, y un tamaño físico máximo de buffer igual a $3.t^{min}$.

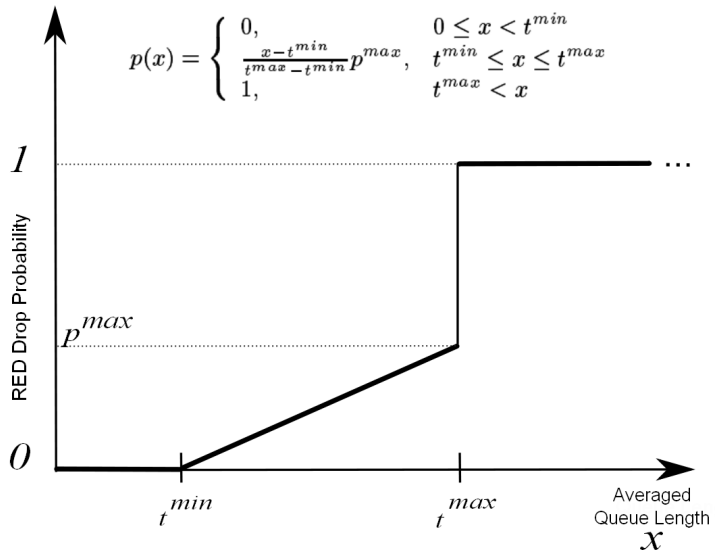


Figura 7.3: Perfil de Probabilidad de Descarte de Paquetes. Mecanismo RED.

7.3. Metodología basada en DEVS

En esta sección se seguirá la metodología presentada en el Capítulo 6 modelando y simulando el sistema bajo estudio en sus formas continua y discreta utilizando DEVS como formalismo unificador, y las nuevas herramientas presentadas en esta Tesis.

Se experimentará con una aproximación fluida del sistema TCP/AQM permitiendo simular sus características dinámicas más relevantes en tiempos muy convenientes. Se hará uso de los nuevos modelos DQSS para modelar el hecho de que la dinámica de la ventana de control de congestión de TCP depende de sus propios valores históricos.

Luego, se desarrollarán nuevos modelos de eventos discretos para simular de manera exacta y detallada el comportamiento de la red bajo el esquema TCP/AQM, usando esta vez primeros principios de los protocolos y medios físicos. Se hará uso de nuevas bibliotecas para la representación detallada de estructuras de paquetes, permitiendo manipularlos de modo individual y diferenciado a lo largo de sus ciclos de vida. Se utilizará también la nueva biblioteca STDEVS para obtener muestras de distribuciones aleatorias y poder representar en detalle tanto la generación de tráfico como las decisiones de control de admisión por parte de RED.

Se desarrollará aquí un modelo a eventos discretos del control de congestión tipo TCP, que representa el primero en su clase desarrollado en DEVS con un grado de detalle considerable.

7.3.1. Controlador Continuo con Retardos. Planta Continua.

Se tomará como caso de estudio el modelo continuo no lineal para TCP/AQM utilizado en [MGT00]. El sistema continuo integrado aproxima las dinámicas de AIMD y de RED por medio de ecuaciones diferenciales, según:

$$\dot{W}(t) = \overbrace{\frac{1}{RTT(q(t))}}^{AI} - \overbrace{\frac{W(t)}{2} \times \underbrace{\frac{W(t-\tau)}{RTT(q(t-\tau))} p(x(t-\tau))}_{RED}}^{MD} \tag{7.1}$$

$$\dot{q}(t) = -\mathbf{1}_{q(t)} C + N \times \frac{W(t)}{RTT(q(t))} \tag{7.2}$$

donde W es el tamaño de la ventana de TCP, RTT es el tiempo de ida y vuelta, q es la longitud de la cola y x es la versión filtrada de q. Se asume que las variables mencionadas representan el valor promedio esperado de sus magnitudes instantáneas.

Este sistema modela un único nodo enrutador de capacidad (ancho de banda máximo) C, utilizado por N usuarios o flujos concurrentes e idénticos (todos experimentan el mismo RTT, y consecuentemente, adoptarán una idéntica ventana W).

El primer término a la derecha de la igualdad en la Ec. (7.1) modela la parte Additive Increase (AI) de AIMD, mientras que el segundo término modela la parte Multiplicative Decrease (MD).

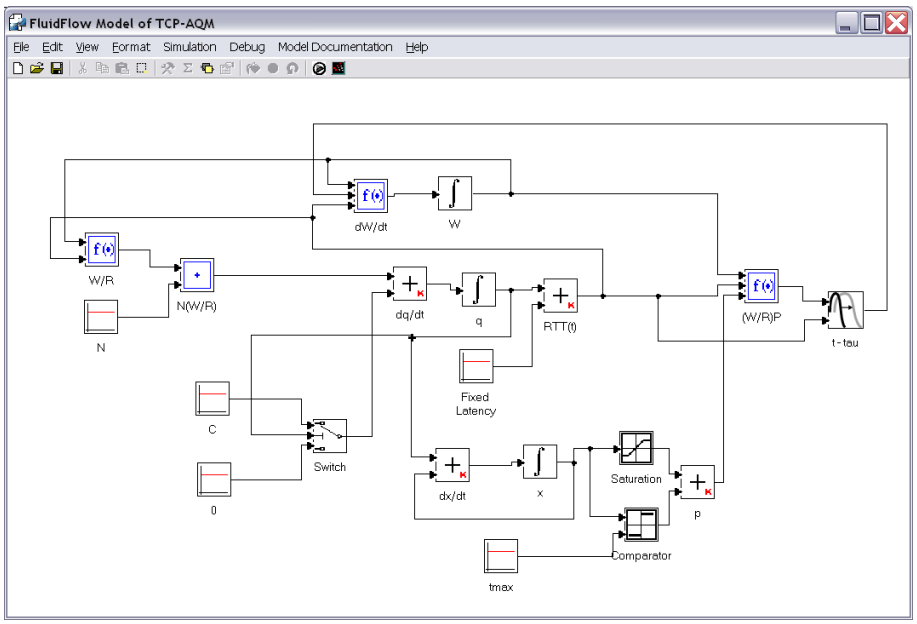


Figura 7.4: Modelo continuo de TCP/AQM implementado en PowerDEVS.

Cabe destacar el factor $\frac{W.p}{RTT}(t-\tau)$ que modula al componente $\frac{W(t)}{2}$. Dicho factor implica: a) que la disminución multiplicativa de $\dot{W}$ dependerá del estado del sistema τ segundos atrás en el tiempo, b) que será función de la tasa efectiva $\frac{W}{RTT}$ de TCP en aquel instante y c) que la probabilidad de que dicha tasa haya producido pérdidas de paquetes estuvo dada por el valor de p en aquel instante. Aún más, se verifica que el retardo temporal τ es variable y coincide con el propio RTT, es decir, $\tau = \tau(t) = RTT(t)$. Esto transforma al sistema Ec. (7.1), (7.2) en una Delay Differential Equation de retardo variable y dependiente del estado, ya que RTT es a su vez función de la variable de estado $q(t)$.

La Ec. (7.2) muestra en el primer miembro a la derecha de la igualdad la tasa C a la cual se vacía la cola del enrutador. El operador $\mathbf{1}_{q(t)}$ toma el valor 1 si la cola no está vacía ($q(t) > 0$), y 0 si lo está. Con este operador tipo llave se previene la situación de alcanzar valores negativos de $q(t)$, lo cual es físicamente imposible. Por su parte, la sumatoria del segundo miembro modela la tasa a la cual se llena la cola del enrutador, aceptando tráfico agregado proveniente de N enlaces idénticos.

El modelado de este sistema realizado en PowerDEVS se muestra en la Figura 7.4. Como se ve en dicha figura, para representar la dinámica retardada de MD se utiliza el bloque de retardo variable $t - tau$, que implementa el método DQSS presentado en el Capítulo 4. El resto de los bloques son parte de la biblioteca de modelos continuos e híbridos de la versión estándar de PowerDEVS. En particular, los bloques integradores se configuran para operar con el método QSS3 (precisión de orden 3).

Para simular este sistema se adoptará el juego de parámetros que se muestra en la Tabla 7.1, inspirados parcialmente en ejemplos provistos en [MGT00]. Por lo pronto se trabajará con $N = 1$, es decir existe 1 único sistema AIMD

Parámetro	Valor	Unidad	Descripción
BW	5E6	bits/seg	Ancho de banda disponible
pkt	500	Bytes	Tamaño de paquete
$C = BW/pkt$	1250	paquetes/seg	Tasa máxima de paquetes
t^{min}	50	paquetes	Umbral inferior de longitud de cola (RED)
t^{max}	100	paquetes	Umbral superior de longitud de cola (RED)
p^{max}	0.1		Probabilidad de descarte para t^{max} (RED)
α	1E-3		Constante de tiempo pasabajos (RED)
$\delta = 1/C$	8E-4		Constante de tiempo pasabajos (RED)
RTT	0.02	seg	Latencia mínima de ida y vuelta

Tabla 7.1: Parámetros de configuración para TCP/AQM

administrando su tamaño de ventana W local.

La simulación se llevó a cabo con un tiempo final $T_f = 100$ seg., y los resultados se muestran en la Figura 7.5. El tiempo de ejecución resultó en 15 seg. promediando algunas repeticiones. Queda evidenciado el cáracter oscilatorio del sistema, corroborando lo esperado acorde a la publicación tomada como referencia para el modelo y sus parámetros.

En primer lugar se verifica el comportamiento esperado para la fase de crecimiento aditivo (AI) de la ventana de congestión W, la cual crece con una pendiente $\dot{W} = \frac{1}{RTT(t)}$ consistente con la evolución de $RTT(t)$ que se muestra en el recuadro ampliado de la Figura 7.5. También se observa el efecto de suavizado de la evolución de la longitud de cola $q(t)$, dado por la señal $x(t)$ que muestra una menor excursión de valores, manteniendo la componente de continua o baja frecuencia. Al observar los instantes en que $p(t)$ se hace distinto de 0, se los puede correlacionar correctamente con las breves fases de decremento multiplicativo (MD) en $W(t)$, lo cual genera: una disminución de la tasa de tráfico enviada por TCP, su correspondiente efecto de disminución en la cola $q(t)$, y una consecuente desaparición de la señal $p(t)$; completando así un ciclo de regulación automática.

En la Figura 7.6 se muestra el detalle del efecto del bloque que implementa el retardo $(t - \tau(t))$, que como se mencionó antes, equivale a $(t - RTT(t))$.

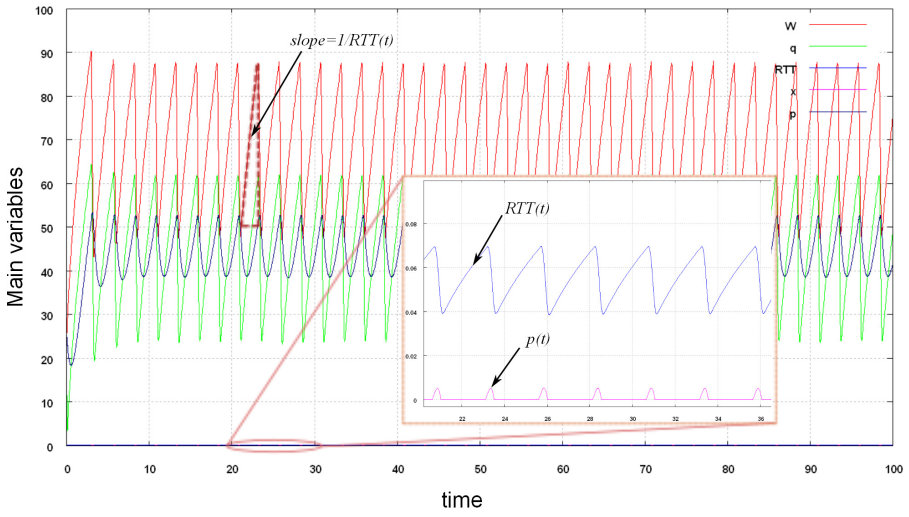


Figura 7.5: Resultados de Simulación. Aproximación Fluida de TCP/AQM con parámetros según Tabla 7.1.

Puede observarse la influencia del carácter variable de $RTT(t)$ sobre la señal que *modula en tiempo*.

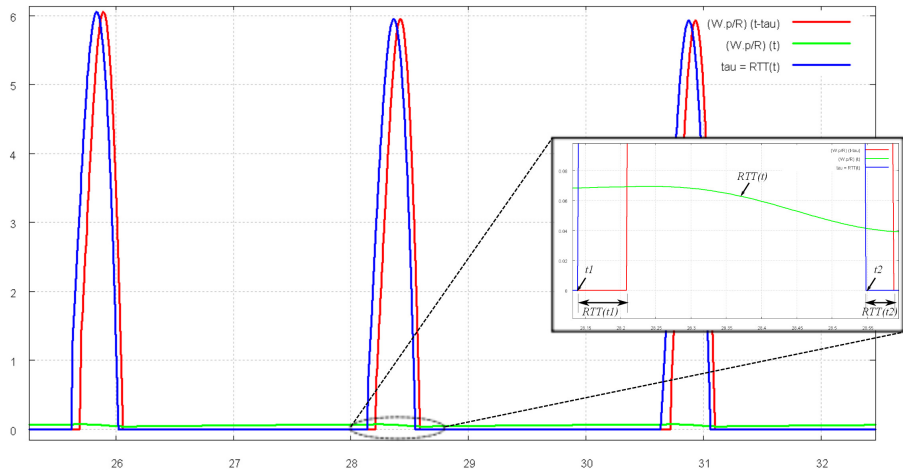


Figura 7.6: Resultados de Simulación. Detalle del efecto del retardo variable calculado por medio del método DQSS.

Los resultados obtenidos permiten investigar de manera rápida y sencilla los efectos ante variaciones de parámetros sobre la dinámica del sistema. Cuando se obtiene una configuración satisfactoria y se pretende analizar en mayor profundidad, es necesario verificar el modelo fluido con una representación más realista y que provea más información.

7.3.2. Controlador y Planta a Eventos Discretos

Continuando con la metodología integradora basada en DEVS, se desarrolla un modelo discreto del sistema TCP/AQM, siguiendo los primeros principios de funcionamiento descritos en las Secciones 7.2.2 y 7.2.1. En la Figura 7.7 se muestra el modelado en PowerDEVS del sistema híbrido a eventos discretos y a tiempo discreto.

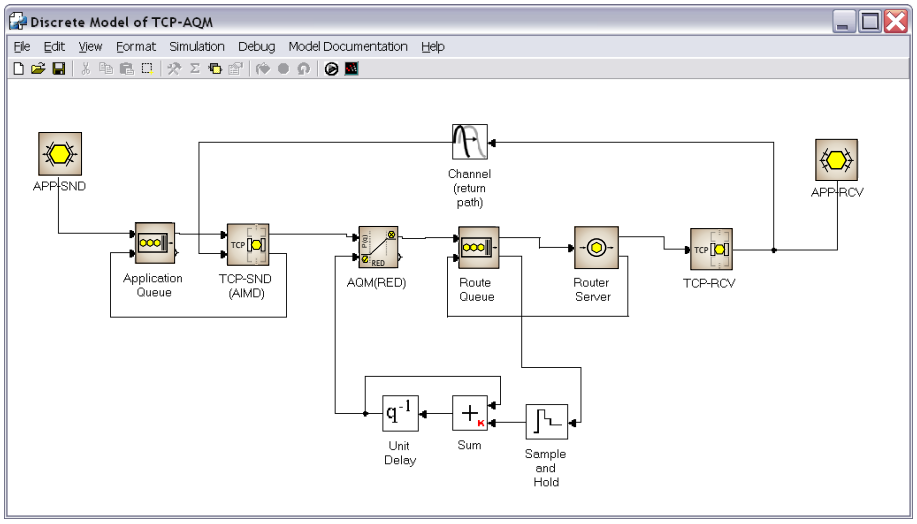


Figura 7.7: Modelo discreto de TCP/AQM implementado en PowerDEVS.

La generación de tráfico desde el bloque **APPSND** (que representa la "aplicación" que usa al sistema TCP/AQM) obedece a una distribución de probabilidad exponencial para el tiempo de inter–generación de paquetes. La decisión de admisión de paquetes por parte de RED obedece a una distribución de probabilidad ad–hoc presentada antes, que se construye a partir de una distribución uniforme en el bloque **AQM(RED)**. En ambas ocasiones se utiliza la nueva biblioteca STDEVS, la cual implementa los generadores provistos por la herramienta GSL - GNU Scientific Library vastamente adoptada por la comunidad científica que desarrolla en C++. El resto de los bloques forma parte de una nueva biblioteca de modelos para PowerDEVS, estando todos ellos capacitados para recibir, emitir y procesar un nuevo formato de mensajes DEVS, que representa paquetes de red con información estratificada de los protocolos que los manipulan.

Nótese que en este modelo se utiliza un bloque de retardo variable continuo, como el utilizado en el modelo continuo de la sección anterior. Esto se explica por la simplificación realizada acerca del canal de regreso de los paquetes de ACK. En lugar de simular exactamente dichos paquetes, lo que se implementa es la notificación del valor de secuencia de ACK, lo que puede transmitirse vía una señal continua. Esta idea disminuye en casi un 50 % la carga computacional de la simulación, ya que de otro modo habría que duplicar el manejo de paquetes "vivos" a cada instante. Con el bloque de retardo se modela una demora constante para los ACK transitando el canal de regreso, suposición que es realista

ya que dicho canal nunca representa un cuello de botella. De este modo se hace uso de la capacidad integradora de la metodología, incorporando una porción continua al sistema por simplicidad de modelo y por conveniencia de simulación.

La simulación se llevó a cabo con un tiempo final $T_f = 30$ seg., y los resultados se muestran en la Figura 7.8. El tiempo de ejecución resultó en 127 seg. promediando algunas repeticiones. Esto arroja una proporción de aproximadamente 28 veces el tiempo de simulación ofrecido por el modelo continuo utilizado en la sección anterior.

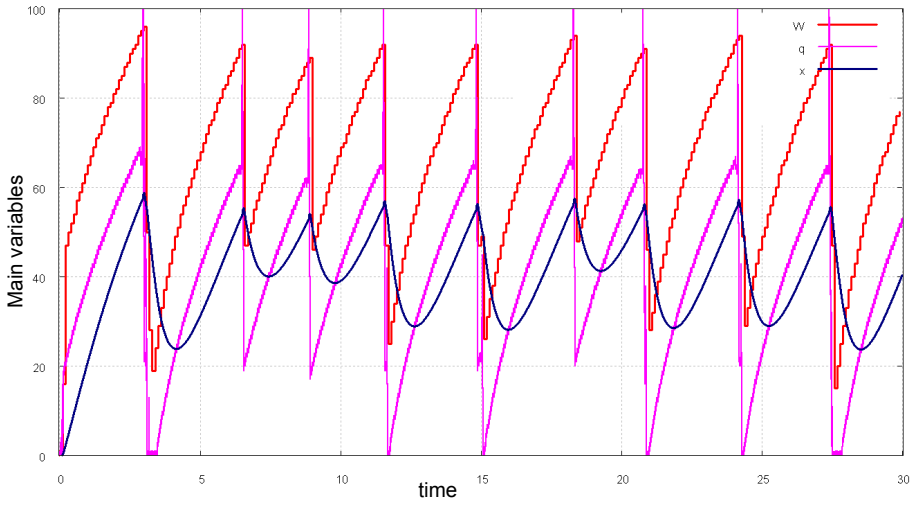


Figura 7.8: Resultados de Simulación. Aproximación discreta de TCP/AQM con parámetros según Tabla 7.1.

Nuevamente se observa un carácter oscilatorio del sistema. Los resultados son cualitativamente comparables, brindando un nivel razonable de confianza sobre la coherencia entre modelos. En la Figura 7.9 se comparan las magnitudes W, q y x para los modelos continuo y discreto durante el intervalo inicial de 30 segundos.

Cualitativamente los rangos de excursión de las magnitudes son satisfactoriamente comparables, considerando que el sistema continuo por definición aproxima evoluciones promediadas que suavizan las altas frecuencias.

Como es de esperarse, por la característica estocástica del sistema discreto se observan evoluciones no homogéneas, con algunas anomalías, que se acercan más al comportamiento real de una red de datos. Por ejemplo, el sistema discreto muestra ocasiones en donde el valor instantáneo de la longitud de cola $q(t)$ cae a cero. Esta descripción es mucho más realista que la evolución observada en la simulación continua.

7.3.3. Experimentos con múltiples flujos idénticos

La característica más sobresaliente de TCP/AQM es su capacidad de adaptar la acción individual de control en cada nodo usuario acorde al estado global de congestión de la red compartida, siendo la congestión una consecuencia del acceso concurrente del conjunto de nodos.

El modelo continuo dado por el sistema Ec. (7.1), (7.2) asume que existen N usuarios idénticos, y por lo tanto calcula una solución para $W(t)$ que representa el comportamiento de AIMD para cada uno de los N flujos que compiten por la red. Intuitivamente, la ventana W en cada nodo debe disminuir a medida que aumenta la competencia por el ancho de banda disponible, con miras a evitar la congestión.

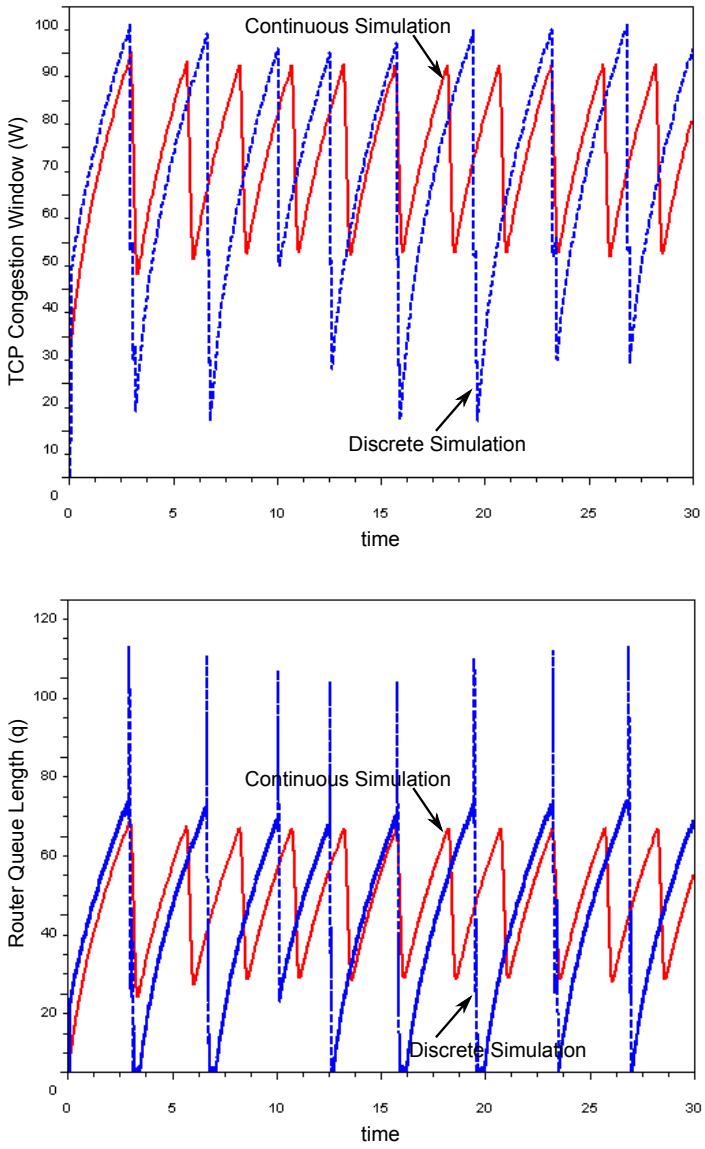


Figura 7.9: Resultados de Simulación. Comparación entre Modelo Discreto y Modelo Continuo.

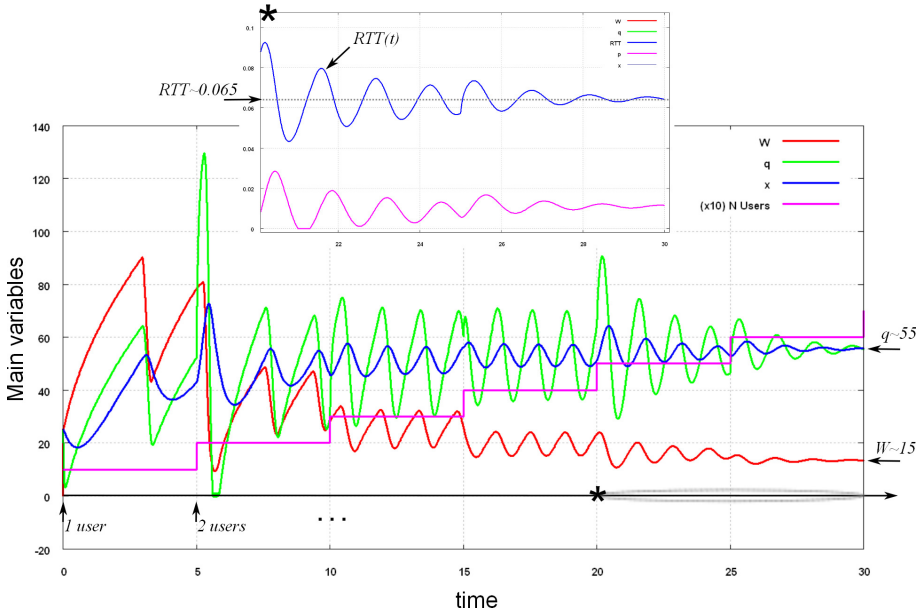


Figura 7.10: Aproximación Fluida de TCP/AQM con ingreso de nuevos usuarios al sistema (desde N=1 hasta N=6).

Para ilustrar esta idea se repite la simulación del modelo continuo de la red suponiendo un ingreso incremental de 1 nuevo usuario cada 5 segundos. El resultado puede observarse en la Figura 7.10. Cabe notar que además de observarse el comportamiento previsto para W, se puede verificar que la longitud de la cola $q(t)$ en el enrutador es regulada satisfactoriamente entorno a un intervalo (en este caso entre los 40 y 60 paquetes, aproximadamente).

Dado que el sistema converge a un aparente equilibrio en su longitud de cola q y tiempo de ida y vuelta RTT puede intentarse una verificación analítica utilizando el resultado clásico de la *Fórmula de Little* [Lit61]. Para un sistema cola–servidor *en equilibrio*, cualquiera sea la distribución de probabilidad del tráfico que lo atraviesa, se verifica $\bar{q} = \lambda_q \bar{R}_q$, en donde $\bar{q}$ es la cantidad de paquetes en la cola, λ es la tasa de ingreso de paquetes a la cola, y $\bar{R}_q$ es el retardo promedio experimentado por cada paquete en entrar y salir de la cola [3]. En el caso del enrutador bajo estudio, en el equilibrio, se tendrá $\bar{q} \approx 55$ paquetes y $\lambda_q = 1250$ paquetes por segundo, para lo cual se deberá verificar $\bar{R}_q \approx 0{,}044$ segundos.

Recordando que la latencia mínima fijada para el camino de ida y vuelta es $RTT = 0{,}02$ segundos, debe verificarse que $RTT(t) = RTT + R_q(t)$ a cada instante, ya que el retardo de encolamiento es la componente variable del tiempo de ida y vuelta. Luego, en el equilibrio deberá verificarse $\bar{RTT} = RTT + \bar{R}_q$. Reemplazando los valores obtenidos por cálculo y por simulación, se verifica satisfactoriamente $\bar{RTT} \approx 0{,}065 \approx 0{,}02 + 0{,}044$.

[3] La notación original es $L = \lambda W$, pero se readapta aquí para evitar confusión con la notación propia de la Tesis.

Análisis Detallado de $W(t)$ Mediante el Modelo Discreto

El experimento con el modelo continuo muestra a $W(t)$ como una representación válida e idéntica para cada usuario de los N presentes en el sistema. Sin embargo esta es una simplificación que omite numerosos detalles de interés acerca del comportamiento discreto y estocástico que sucede en una red.

Para tener una percepción más realista de la dinámica concurrente de W en cada nodo, se recurre nuevamente al modelo discreto, esta vez, modelando $N = 6$ conexiones TCP que comparten un único enrutador. Se activará el ingreso de una nueva conexión cada 5 segundos, para imitar el experimento realizado con el modelo continuo. En la Figura 7.11 se observa el resultado de la simulación durante 30 segundos.

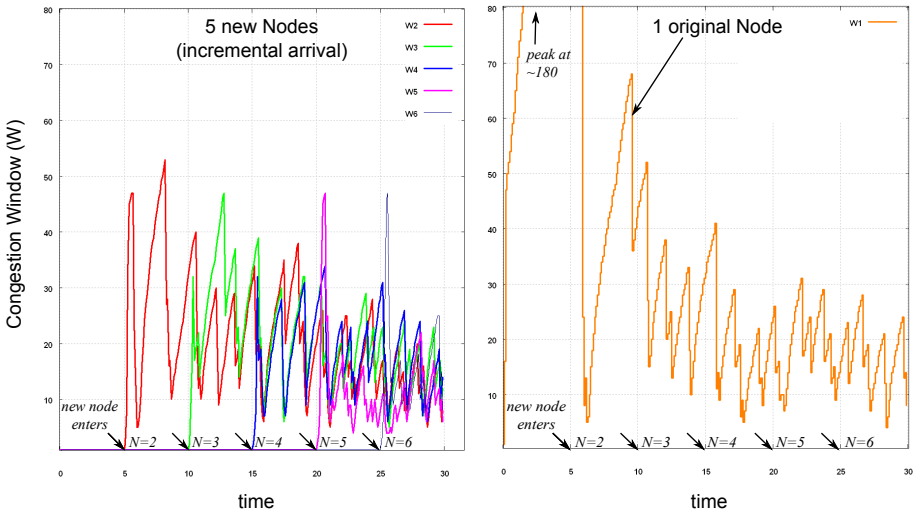


Figura 7.11: Aproximación Discreta de TCP/AQM con ingreso de nuevos usuarios al sistema (desde N=1 hasta N=6).

A la derecha se muestra la evolución del usuario que ingresa al sistema desde el comienzo. A la izquierda se muestra el ingreso cada 5 segundos de un nuevo usuario con idénticas características (es decir, sometido al mismo retardo $RTT(t)$). Las curvas W_2 a W_6 muestran en forma detallada la adaptación a las nuevas condiciones de congestión que va sufriendo W en cada nodo. Por ejemplo, puede notarse que en los instantes de ingreso de cada nuevo usuario, se produce una congestión tal que señaliza prontamente la situación a todos los nodos en el sistema. Por ello, todas las ventanas W caen a la mitad de sus valores de modo casi sincronizado. Esta sincronización no se observa durante los períodos de régimen estable en donde no ocurren cambios estructurales en la red.

Como primer observación respecto del modelo continuo surge la discrepancia entre la evolución de cada W_n discreta (Figura 7.11) y su correlativa W continua (Figura 7.10) entre los tiempos de simulación $t_s = 25$ y $t_s = 30$, es decir, cuando hay $N = 6$ usuarios en el sistema. Para el modelo discreto, $W_n(t)$ permanece oscilando entre los valores 5 y 25 aproximadamente, con un valor promedio de 15. En cambio, en el modelo continuo $W(t)$ converge hacia el equilibrio, también con un valor promedio de 15. En párrafos anteriores se realizó un análisis basado en

teoría de colas asumiendo válida esta condición de equilibrio aparente para $W(t)$. Si bien el análisis permanece válido para el modelo continuo, la experimentación con el modelo discreto invalida la representatividad del primero para $N = 6$ en tanto desaparece la dinámica oscilatoria, produciendo un equilibrio artificial.

Inmediatamente se pudo comprobar que la constante de tiempo empírica $\delta = 1/C$ tomada de la literatura para el filtro pasabajos que suaviza la evolución de la medición de $q(t)$, tiene incidencia directa sobre la desaparición no deseada de las oscilaciones. Luego, y también de forma empírica, se modificó dicho parámetro haciéndolo función de N, del siguiente modo: $\delta = \frac{1}{NC}$. Con este cambio, se mantuvieron idénticas las evoluciones para N menores a 6 y se logró que el modelo continuo reproduzca las oscilaciones para $N = 6$.

Este hecho pone de relevancia la importancia de la metodología integradora propuesta en esta Tesis, basada en un formalismo unificador y matemáticamente robusto. Si se estuviese trabajando con distintas herramientas de software y distintos algoritmos para el modelado y la simulación de los sistemas continuo y discreto, existiría una gran incertidumbre a priori acerca del origen de discrepancias como las descritas en el párrafo anterior. Con la metodología integradora basada en DEVS, haciendo uso de la teoría presentada en esta Tesis justificando la validez de la representación de sistemas estocásticos, por un lado, y la correctitud de la aproximación de ecuaciones diferenciales con retardo, por el otro, puede tenerse una alta certeza de que la teoría y la implementación del simulador –tratamiento del avance del tiempo, representación de los datos, etc.– son absolutamente coherentes y no son causantes de anomalías. Evidentemente, como sucede con cualquier metodología basada en modelado y simulación, los problemas aún pueden surgir tanto por la formulación deficiente de los modelos y/o la comisión de errores humanos al implementarlos.

Desafortunadamente, la solución encontrada deja de funcionar nuevamente cuando se hace $N = 8$, evidenciando que se requiere un estudio más detallado acerca del modelo continuo y su validez para N creciente. Para el caso planteado (y en un primer análisis) se puede adoptar como válida la conclusión temporaria de que el modelo continuo así parametrizado queda verificado por el modelo discreto hasta $N = 6$. A los efectos de la presente Tesis y la postulación de la metodología unificadora de modelado y simulación, será suficiente trabajar dentro de dicho rango de validez.

7.4. Modelado Híbrido Continuo/Discreto

Como se mencionara en la Sección 2.5 existen enfoques que intentan combinar la ventaja de la eficiencia computacional de una simulación fluída con la ventaja del detalle granular provisto por una simulación discreta (ver [SLCP01, GLT04, YS07] y las referencias provistas allí). Esta técnica se conoce como modelado y simulación de flujos híbridos, en donde el concepto de hibridez se refiere a la representación heterogénea y simultánea del *un mismo fenómeno*, distinguiéndose del concepto más general de hibridez que hace referencia a la coexistencia de *distintos fenómenos* heterogéneos en un mismo modelo.

Los antecedentes de la literatura que modelan flujos híbridos muestran enfrentar serios desafíos surgidos esencialmente de problemas de sincronización entre el manejo del paso de integración (propio del método numérico elegido para aproximar la parte fluida) y el manejo del avance de tiempo virtual para

los eventos discretos. Diversas técnicas se proponen para modificar los motores de simulación a eventos discretos de propósito específico (e.g., ns–2, Opnet) encapsulando en ellos métodos de integración numérica o bien construyendo interfaces hacia solvers externos. Si bien estos enfoques muestran resultados empíricos aceptables, surgen de iniciativas pragmáticas cuya validez está acotada a cada herramienta en cuestión, y que carecen de un marco matemático formal para garantizar su correctitud.

En la presente sección se mostrará como es posible implementar una estrategia sencilla de flujos híbridos de manera casi directa basándose en DEVS. Las propiedades formales de legitimidad y clausura bajo acoplamiento de DEVS (preservadas por STDEVS para procesos estocásticos), sumadas a las propiedades de convergencia, estabilidad y tratamiento eficiente de discontinuidades de QSS y DQSS (para aproximar sistemas continuos), permiten obtener una solución híbrida bajo un formalismo matemático unificador, y en una misma herramienta.

7.4.1. Integración de Flujos Continuos y Discretos en DEVS

La idea intuitiva consiste obtener un modelo híbrido del flujo total de paquetes del enrutador, en el cual se represente simultáneamente una parte continua (modelando el efecto de N_{cont} usuarios) y otra parte discreta (modelando el efecto de N_{disc} usuarios). Ambas partes deben influírse mutuamente para representar fielmente la dinámica de la totalidad de los usuarios N_{tot}.

Se propondrá aquí una estrategia que consiste en asignar la totalidad de los usuarios a la parte continua ($N_{tot} = N_{cont}$) ubicando así al sistema en su estado de régimen, y en simultáneo mantener un único usuario discreto ($N_{disc} = 1$) a modo de *flujo testigo discreto*, permitiendo observar detalles con la granularidad que se desee. Siguiendo esta idea, se adopta la suposición $N_{cont} \gg N_{disc}$, que en principio permitirá despreciar la influencia de la parte discreta sobre la parte continua. Luego, sólo resta determinar la forma en que el flujo testigo discreto es influído por el estado del sistema continuo.

Se elegirá al sistema cola–servidor como aquella parte sistema que adquiera las características híbridas necesarias para modelar la influencia de la parte continua sobre la parte discreta, del modo que se describirá a continuación.

Informalmente, una cola híbrida debe representar el estado de llenado de la parte continua $q_c(t)$ instante a instante, manteniéndolo consistente con el estado de la parte discreta que modela la llegada, ubicación, desplazamiento y partida de paquetes individuales, y su cantidad total $q_d(t)$ instante a instante.

Un esquema descriptivo de esta idea puede encontrarse en la Figura 7.12.

Ante el arribo de un nuevo paquete discreto a la cola híbrida, el mismo debe "ver"delante de sí un número de paquetes equivalente al total presente en el sistema, es decir, debe encontrarse con los $q_c(t)$ paquetes indicados por el sistema continuo. Surge entonces la idea de construir un paquete híbrido artificial por cada paquete discreto, de forma tal que su longitud equivalente haga cierta la igualdad $q_c(t) = q_d(t) + q_h(t)$ en todo momento. Se define a $q_h(t)$ como la longitud híbrida artificial de cola que representa la cantidad de paquetes continuos que deberían intercalarse entre paquetes discretos adyacentes, alcanzando así la longitud total $q_c(t)$. Para que este mecanismo funcione, el server deberá descargar de la cola paquetes híbridos, y para cada uno de ellos, simular un tiempo

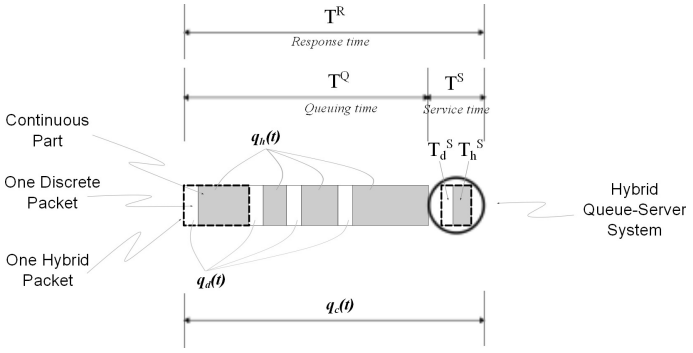


Figura 7.12: Sistema Híbrido Cola–Servidor. Paquetes discretos (en blanco), Paquetes equivalentes continuos (en gris) y Paquetes híbridos (en recuadro a rayas).

de servicio correspondiente a 1 paquete discreto (T_d^S) más el tiempo de servicio artificial equivalente (T_h^S), correspondiente a la cantidad extra de paquetes que se le adicionó al ingresar a la cola.

Para formalizar esta idea, se analizará un sistema cola–servidor continuo puro, otro discreto puro, y luego se calculará la proporción de paquetes artificiales a adicionar para componer cada paquete híbrido. Como referencia para el cálculo, en la Figura 7.12 se muestran el Tiempo de Encolamiento T^Q, Tiempo de Servicio T^S y Tiempo de Respuesta $T^R = T^Q + T^S$ que definen las demoras principales que tienen lugar en el sistema cola–servidor operando a una capacidad de C paquetes por segundo.

Para el caso continuo, considerando al sistema completo en su aproximación fluida y en equilibrio, se recurre nuevamente a la Fórmula de Little:

$$C.T_c^R = q_c \tag{7.3}$$

en donde T_c^R es el tiempo de respuesta del sistema cola–servidor y q_c es la cantidad de elementos en dicho sistema, ambos para el caso continuo. Si se asume que todos los paquetes son del mismo tamaño y que en la aproximación fluida siempre existen paquetes para procesar, el tiempo de servicio será $T_c^S = \frac{1}{C}$. Reemplazando en la ecuación anterior, se verifica:

$$T_c^R = q_c.T_c^S \tag{7.4}$$

que relaciona el tiempo de respuesta total, el estado de llenado y el tiempo de servicio (en sus valores promedio) para el sistema continuo.

Por su parte en el sistema discreto, manteniendo la suposición de que todos los paquetes tienen la misma longitud, el tiempo de servicio T_d^S será constante e igual para todos. Luego, deberá verificarse:

$$T_d^R = T_d^S.(1 + q_d) \tag{7.5}$$

en donde T_d^R es el tiempo de respuesta y $q_d + 1$ es la cantidad de elementos en todo el sistema (que incluye al que está siendo procesado en el Server), ambos para el caso discreto.

Acorde a las características sugeridas para el sistema híbrido, deberá verificarse:

$$T_h^R = T_c^R \tag{7.6}$$

es decir, el sistema híbrido y el sistema continuo deben ser equivalentes desde un punto de vista externo imponiendo el mismo retardo (tiempo de respuesta total) al flujo de paquetes circulante. Otra característica que deberá verificarse es la siguiente:

$$T_h^R = T_h^S(1 + q_d) \tag{7.7}$$

es decir, el sistema híbrido y el sistema discreto deben ser equivalentes desde el punto de vista de la cantidad de paquetes (híbridos y discretos) que hay en la cola, lo que explica el uso de q_d en la ecuación anterior. El tiempo de servicio T_h^S de la parte híbrida representa artificialmente parte del estado de la cola continua.

Finalmente, reemplazando las Ec. (7.7) y (7.4) en la Ec. (7.6) y reordenando, se obtiene el tiempo de servicio de la parte híbrida de los paquetes que deberá calcular el nuevo sistema ante la llegada de cada paquete discreto:

$$T_h^S = T_c^S \frac{q_c}{1 + q_d} \tag{7.8}$$

La suposición realizada para la parte continua de que todos los paquetes tienen igual longitud debe ser consistente con la parte discreta, para poder integrarlas luego en el sistema híbrido. Es decir, el tiempo de servicio de un paquete individual deberá ser el mismo en ambas representaciones, verificando: $T_c^S = T_d^S$. Con esta consideración, y definiendo $\Upsilon = \frac{q_c}{1+q_d}$ como factor de proporcionalidad, se obtiene finalmente:

$$T_h^S = T_d^S \frac{q_c(t)}{1 + q_d(t)} = T_d^S.\Upsilon(t) \tag{7.9}$$

7.4.2. Implementación del Bloque Combinador Híbrido de Flujos

.

Desde el punto de vista de la implementación, cada paquete discreto trae consigo la información de su longitud, y de ella entonces puede determinarse inmediatamente el tiempo de servicio T_d^S por medio del factor Υ. Luego, el nuevo requisito para construir el sistema cola–servidor híbrido es que dicho sistema tenga conocimiento de $\Upsilon(t)$ en los instantes de arribo de cada paquete discreto. Dicha información es suficiente para convertir a cada paquete discreto en el nuevo paquete híbrido correspondiente.

Para implementar un sistema cola–servidor híbrido compatible con los modelos DEVS obtenidos previamente para los bloques *Queue* y *Server*, se desarrolló el modelo *HybridFlow* que combina flujo discreto y continuo, evitando realizar modificaciones a las bibliotecas existentes.

En la Figura 7.13 se muestra esta idea implementada en la herramienta PowerDEVS, en donde se antepone el nuevo bloque HybridFlow a una cascada formada por Queue y Server construida con bloques previamente desarrollados (utilizados a lo largo de los casos de estudio discretos de esta Tesis).

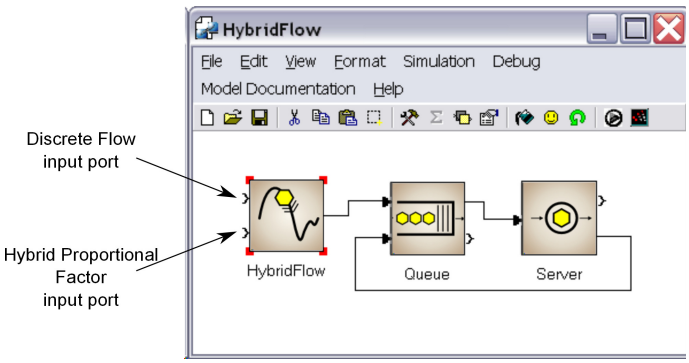


Figura 7.13: Nuevo Bloque HybridFlow para combinar flujos discretos y continuos.

El primer puerto de entrada (superior) acepta paquetes discretos y el segundo puerto de entrada (inferior) recibe una señal continua que representa $\Upsilon(t)$. En el puerto de salida se emiten paquetes híbridos con una longitud modificada tal que cuando el paquete llegue al Server, el tiempo de servicio sea el definido por la Ecuación 7.9. Cuando el paquete sale del server y continúa avanzando por el sistema discreto, lo hará preservando su longitud discreta original. Esto es, un paquete tendrá carácter híbrido sólo desde que ingresa a un bloque HybridFlow y hasta que egresa de un bloque Server.

7.4.3. Modelo Híbrido de TCP/AQM

Se aplicará la técnica híbrida descrita en esta sección al caso de estudio TCP/AQM presentado en 7.3.1 y 7.3.2, con el propósito de evaluar su eficacia para disminuir el tiempo de simulación del sistema discreto a la vez que provea información detallada de un flujo discreto testigo. Una ventaja adicional que se espera obtener del modelado híbrido es la practicidad en el modelado visual con la herramienta PowerDEVS. Cuando el número de usuarios individuales crece, el modelado visual explícito de cada conexión se torna engorroso, como puede observarse en la Figura 7.14 para el caso de N=6 usuarios (compárese con la Figura 7.7 utilizada antes para N=1).

En cambio, utilizando la técnica híbrida propuesta, sólo es necesario modelar explícitamente un único usuario discreto (representando el flujo testigo), valiéndose de la parte continua del modelo para llevar al sistema al régimen de operación correspondiente para N=6. Este enfoque se muestra implementado en PowerDEVS en la Figura 7.15.

Se analizarán comparativamente los resultados de simulación para los modelos continuo puro, discreto puro e híbrido, para N=2, 4 y 6. Se hará foco sobre la variable de estado $W(t)$ representativa del tamaño de ventana de TCP (equivalente para cada uno de los usuarios). Los resultados que se presentan en la Figura 7.16 muestran ser satisfactorios desde un punto de vista cualitativo, evidenciando una adecuación consistente en amplitud y frecuencia entre los 3 tipos de modelos ensayados.

Los tiempos de simulación resultantes se muestran en la Figura 7.17, confir-

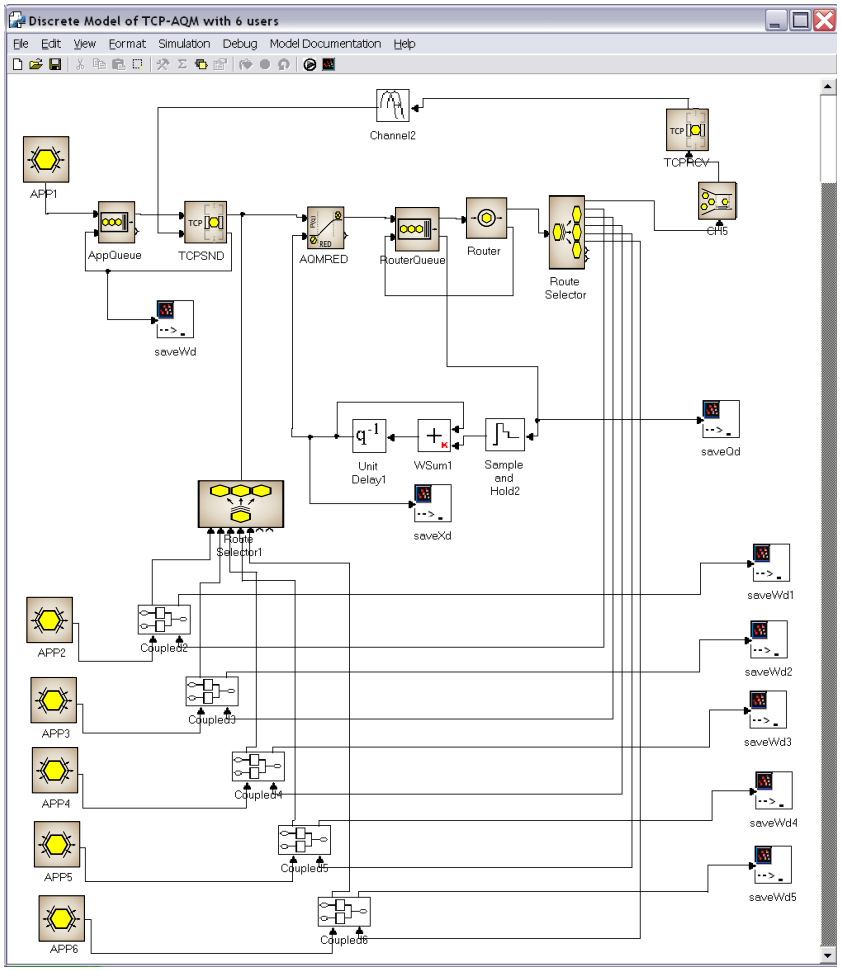


Figura 7.14: Modelo Discreto exacto de TCP/AQM para N=6 usuarios

mando parcialmente las expectativas. Por un lado los tiempos de ejecución del modelo continuo (aproximación fluída) mostraron mejoras (speedups) dentro de un rango de 20 a 30 veces sobre la simulación discreta exacta. Por otro lado, al utilizar el modelo híbrido la ventaja anterior se ve afectada reduciendo los speedups a un rango aproximado entre 1.5 y 2.5 veces, mejorando de todos modos los tiempos de ejecución del sistema discreto puro. La tendencia muestra que al aumentar N esta ventaja se va incrementando. Esta tendencia se verificó para N=8 y N=10, aunque dichos resultados deben ser tomados con cautela al comparar las mejoras entre el modelo continuo y discreto, ya que el continuo adolece de la convergencia artificial al equilibrio mencionada en la sección anterior para $N > 6$.

Los resultados mostrados verifican la validez cualitativa y mejora muy aceptables provistas por la estrategia de modelado híbrido presentada en esta sección. Una mejora colateral evidente es la sencillez de modelado visual que ofrece un sistema híbrido frente a su correlativo discreto cuando el número de usuarios N

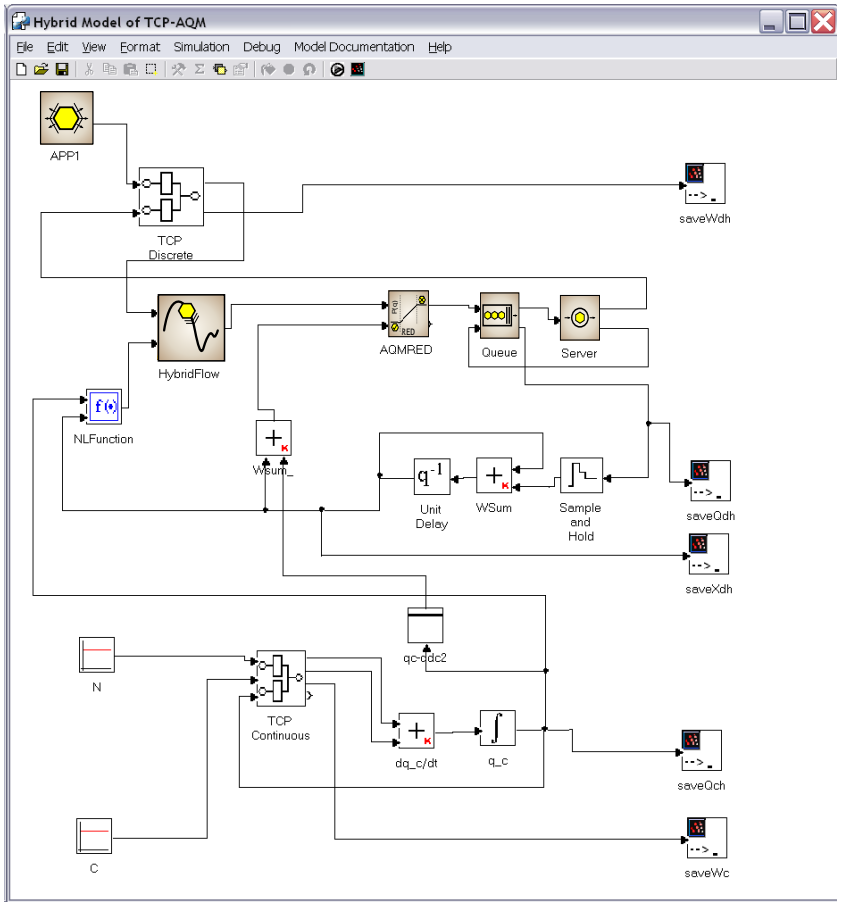


Figura 7.15: Modelo híbrido Continuo/Discreto de TCP/AQM

crece.

Restan llevar a cabo estudios que expliquen el comportamiento de estas mejoras para hacerlas más predecibles, y eventualmente poder dar una estimación analítica del speedup ofrecido por cualquier sistema híbrido. Intuitivamente, para un modelo híbrido con N dado, al conocer el tiempo de ejecución de su equivalente continuo puro (con N usuarios), y el tiempo de ejecución para su equivalente discreto puro (con N=1 usuario), debería poder establecerse el tiempo de ejecución híbrido como aproximadamente la suma de los dos anteriores. Esta relación no queda claramente evidenciada en los experimentos realizados, indicando la necesidad de realizar futuros análisis exhaustivos al respecto.

7.5. Conclusiones

En este capítulo se logró aplicar eficientemente PowerDEVS junto con los nuevos métodos DQSS y la teoría STDEVS para modelar tanto una representación continua del Control de Congestión tipo TCP/AQM como una especifi-

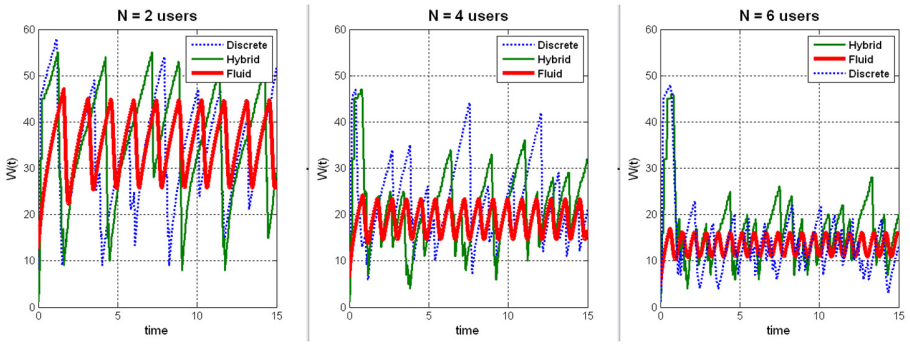


Figura 7.16: Resultados de simulación para modelos Discreto puro, Continuo puro e Híbrido (escenarios para N=2,4 y 6 usuarios concurrentes.

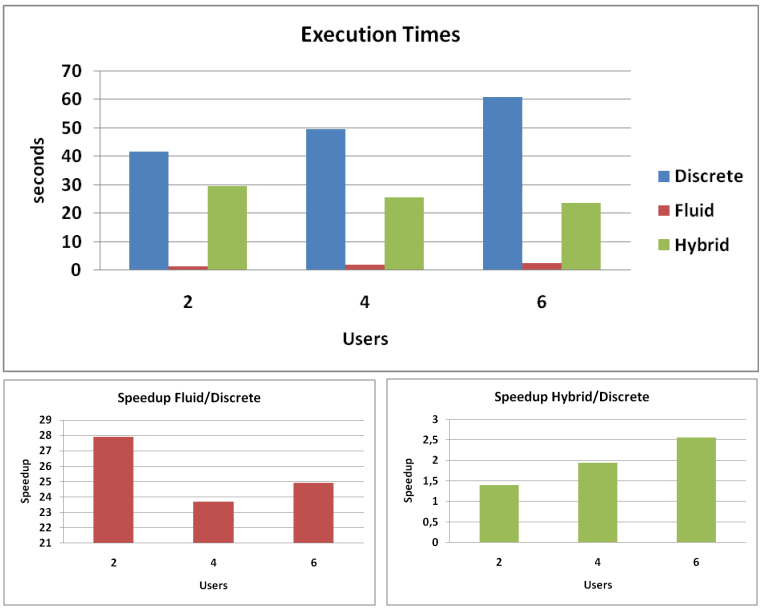


Figura 7.17: Mejoras en Tiempos de Ejecución de simulaciones para escenarios de N=2,4 y 6 usuarios concurrentes (Tiempo de Simulación final = 15 segundos.)

cación discreta.

Además, se propuso una nueva metodología para el modelado y simulación de flujos híbridos que permite combinar simultáneamente las ventajas de las representaciones fluída y discreta. Se desarrolló un nuevo modelo Combinador Híbrido de Flujos que hace posible la obtención de un flujo híbrido a partir de un flujo discreto y una señal continua. Dicho modelo fue aplicado exitosamente para disminuir los tiempos de simulación discreta de TCP/AQM preservando detalles granulares a nivel de paquetes individuales.

DEVS proveyó el marco formal integrador que es el objetivo que se pretendía alcanzar, habilitando a seguir un proceso iterativo de refinamiento de

modelos y adoptando las aproximaciones continuas y/o discretas que se crea conveniente para cada necesidad de diseño. Esto es posible gracias al marco matemático coherente que aporta DEVS y las nuevas herramientas desarrolladas en la presente Tesis, especificando sin ambigüedades las relaciones estructurales y de manejo del tiempo indispensables para garantizar la compatibilidad entre los distintos paradigmas de representación.

Capítulo 8

Conclusiones Generales y Problemas Abiertos

Se presentó un nuevo formalismo para describir sistemas de eventos discretos estocásticos. Basado en el enfoque de Teoría de Sistemas de DEVS y haciendo uso de la teoría de Espacios de Probabilidad, STDEVS provee un marco formal para modelado y simulación para sistemas de eventos discretos genéricos no deterministas. STDEVS consiste entonces en un formalismo universal de modelado que provee la capacidad de representar comportamiento estocástico sobre espacios medibles de dimensión infinita, junto con la capacidad de representar formalismos estocásticos tradicionales vastamente utilizados en diversas aplicaciones prácticas (por ejemplo, Cadenas de Markov, Redes de Colas, Redes de Petri Estocásticas). Debido a su raíz basada en DEVS, STDEVS es también una representación basada en Teoría de Sistemas, proveyendo ventajas específicas de modelado interdisciplinario para especificar sistemas híbridos.

Se presentó una nueva clase de algoritmos de integración para Ecuaciones Diferenciales con Retardo (DDE) y se mostró su aplicación por medio de un número de problemas de la literatura, estudiando su desempeño. Los nuevos métodos DQSS se basan en la cuantificación de las variables de estado en lugar de discretizar el tiempo, ofreciendo algunas propiedades relevantes muy interesantes que los hacen particularmente atractivos. Se demostró que un sistema DDE lineal invariante en el tiempo y asintóticamente estable con retardos constantes produce una aproximación QSS equivalente que permanece numéricamente estable para cualquier quantum ΔQ. La granularidad de la cuantificación influye en la precisión de los resultados de simulación, pero no afecta a la estabilidad numérica. También se demostró que un sistema DDE de tipo ISS (por *Input to State Stable*) no lineal y variable en el tiempo, con retardos variables (e incluso dependientes de los estados) produce una aproximación QSS que permanece numéricamente estable para cualquier quantum suficientemente pequeño ΔQ. Para cualquier sistema de este tipo, existe un quantum $\Delta Q_{\max}$ mayor que cero tal que su aproximación QSS permanece numéricamente estable para todo $\Delta Q \leq \Delta Q_{\max}$. Mientras los algoritmos de control de paso usados en métodos de integración por discretización del tiempo usualmente controlan el error de integración local, los algoritmos basados en cuantificación de estado controlan el error de integración global. Luego, estos últimos proveen mayor robustez que sus

competidores de tiempo discreto. Finalmente, el código de DQSS implementado en PowerDEVS, es mucho más sencillo de utilizar que dde23 y que dde_solver, gracias a la interfaz de usuario gráfica intuitiva que provee PowerDEVS (similar a la provista por Simulink).

Se logró eludir los problemas que se presentan al tratar con sistemas híbridos debido a la necesidad de recurrir a diversos paradigmas y herramientas de modelado y simulación. Esto se hizo posible gracias a la metodología integral propuesta, que se basa en DEVS para aplicar teoría de control a sistemas de admisión de paquetes en redes de datos utilizando modelos híbridos estocásticos.

Se combinaron sistemas continuos con sistemas de eventos discretos en un marco unificado apto para integrar disciplinas heterogéneas en el área de control de sistemas de cómputo y/o red en tiempo real. Se logró la transición directa entre herramientas basadas en DEVS, facilitando la implementación de modelos embebidos en un procesador de red Intel IXP2400, reduciendo efectivamente los riesgos y esfuerzos de recodificación.

Problemas Abiertos

Existen algunos problemas abiertos para continuar con el desarrollo de herramientas basadas en el formalismo DEVS para modelar y simular sistemas dinámicos híbridos.

Esta Tesis presentó nuevas técnicas de simulación para sistemas continuos e híbridos basados en la aproximación de las partes continuas mediante la cuantificación de las variables de estado (métodos de integración por cuantificación) que muestran grandes ventajas computacionales en la integración numérica de sistemas con discontinuidades y en sistemas con retardos. También se presentó una metodología de modelado y simulación para sistemas de eventos discretos estocásticos.

Un posible próximo paso consiste en combinar ambas metodologías generando nuevas técnicas de simulación aplicables a sistemas que incluyan partes discretas y continuas estocásticas.

Por lo tanto, desde el punto de vista teórico se trabajará en el desarrollo de Sistemas de Estados Cuantificados (QSS) para ecuaciones diferenciales estocásticas y en el estudio de sus principales propiedades analíticas. Desde el punto de vista de la aplicación, siguiendo la línea de esta Tesis, se trabajará principalmente en problemas de modelado, simulación y control de redes de datos y más generalmente en sistemas de tipo cola–servidor (sistemas de cómputo de alta performance, por ejemplo).

Como se ha dicho antes, la interacción entre las variables continuas y discretas de los sistemas híbridos provoca discontinuidades en las ecuaciones diferenciales. Estas discontinuidades causan muchas dificultades a los algoritmos de integración numérica ya que los mismos deben detectar exactamente los instantes en los que se producen y reiniciar la simulación a partir de dichos instantes [CK06], lo que conlleva un aumento muy importante de los costos computacionales.

Un problema similar experimentan los métodos de integración en presencia de Ecuaciones Diferenciales con Retardo (DDE), lo que ha motivado el desarrollo de algoritmos numéricos especiales [WB92, ST01]. Los sistemas híbridos con retardos, a su vez, combinan ambas dificultades [S.P08].

La presencia de componentes estocásticas agrega nuevas dificultades a estos problemas. Si bien la generación de secuencias pseudo–aleatorias con buenas propiedades estadísticas no es un problema en este contexto [Hig01], la conservación de dichas propiedades tras la aproximación numérica es un problema crítico que se suma a las dificultades de la integración numérica en sí. Si bien muchos de los métodos para integrar ecuaciones diferenciales estocásticas (SDE) han sido derivado de los métodos clásicos de integración para casos deterministas [Rüm82, GSH88], sus propiedades difieren en función de la convergencia que muestran respecto a los estadísticos. Además, el estudio de la estabilidad numérica de los algoritmos de integración de SDEs es mucho más complejo que en el caso de las Ecuaciones Diferenciales Ordinarias (ODE) [SM96]. Como en las ODE, la presencia de retardos y discontinuidades en las SDEs agrega una gran complejidad adicional [BB00, Buc00, HMY07, ZSL09].

El formalismo DEVS permite representar cualquier sistema discreto (incluyendo aproximaciones numéricas de sistemas continuos) y luego de esta Tesis permite la representación general de sistemas discretos estocásticos [CKW10]. La formalización de los modelos DEVS estocásticos (STDEVS) abre las puertas a la utilización de los métodos de cuantificación de estados para integrar ecuaciones diferenciales estocásticas; ya que en principio estos algoritmos transformarían las SDEs en modelos STDEVS.

Desde el punto de vista de las aplicaciones, en la literatura se ha propuesto una variedad de modelos basados en SDEs no lineales para estudiar la dinámica de TCP [PFTK98, Kel01, GM02, Mas05] en función de la probabilidad de pérdida de paquetes, la incertidumbre del tiempo de retardo entre nodos, y las particularidades estocásticas del tráfico (no estacionariedad, ráfagas, autosimilaridad, etc. [PW00]). Un área abierta de investigación consiste en analizar la robustez de estos métodos para manejar topologías de red con múltiples puntos de congestión [GM04] y estudiar las garantías de escalabilidad que ofrecen para enfrentar el crecimiento exponencial de Internet.

Una aplicación relacionada de particular potencial para aplicar modelado y simulación híbrido involucrando SDEs es el estudio combinado de la performance de protocolos de comunicación estándar (TCP/IP, UDP/IP, RTP, etc.) cuando se los somete a canales de comunicación inalámbricos. El modelado de las características físicas electromagnéticas (que inducen demoras, corrupción y pérdidas en los flujos de paquetes) suele involucrar SDEs (con o sin retardos) [KLP04] para representar antenas, canales dispersivos, entornos ruidosos, etc. Es de particular interés la combinación de los modelos de estos fenómenos continuos con los modelos de eventos discretos que permitan simular el comportamiento detallado de los flujos de paquetes de información.

Bibliografía

[ABS06] M. Bozga A. Basu and J. Sifakis. Modeling heterogeneous real-time components in bip. In *Proceedings of SEFM 2006*, New York, NY, USA, 2006.

[Agg75] S. Aggarwal. *Ergodic Machines - Probabilistic and Approximate Homomorphic Simplifications*. PhD thesis, The University of Michigan, Ann Arbor, Michigan, 1975.

[ALLY02] S. Athuraliya, S.H. Low, V.H. Li, and Q. Yin. REM: Active queue management. *Network, IEEE*, 15(3):48–53, 2002.

[AMBC+95] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, 1995.

[ARH+06] Karl-Erik Arzen, Anders Robertsson, Dan Henriksson, Mikael Johansson, Hakan Hjalmarsson, and Karl Henrik Johansson. Conclusions of the ARTIST2 roadmap on control of computing systems. *SIGBED Rev.*, 3(3):11–20, 2006.

[Bar02] C. Barakat. TCP/IP modeling and validation. *Network, IEEE*, 15(3):38–47, 2002.

[BB00] C. T. H. Baker and E. Buckwar. Numerical Analysis of Explicit One-Step Methods for Stochastic Delay Differential Equations. *LMS Journal of Computation and Mathematics*, 43:315–335, 2000.

[BBM03] F. Budinsky, S.A. Brodsky, and E. Merks. *Eclipse modeling framework*. Pearson Education, 2003.

[BGK+06] K. Balasubramanian, A. Gokhale, G. Karsai, J. Sztipanovits, and S. Neema. Developing applications using model-driven design environments. *Computer*, 39(2):33–40, 2006.

[BHLO03] S. Bohacek, J. Hespanha, J. Lee, and K. Obraczka. A hybrid systems modeling framework for fast and accurate simulation of data communication networks. In *Proceedings of the 2003 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, pages 58–69, New York, NY, USA, 2003. ACM.

[Bil95] Patrick Billingsley. *Probability and Measure*. Wiley, New York, NY, 1995. Third Edition.

[BK10] F. Bergero and E. Kofman. PowerDEVS. A Tool for Hybrid System Modeling and Real Time Simulation. *Simulation: Transactions of the Society for Modeling and Simulation International*, 2010. in press.

[BKBZ08] F. Bergero, E. Kofman, C. Basabilbaso, and J. Zúccolo. Desarrollo de un simulador de sistemas híbridos en tiempo real. In *Proceedings of AADECA 2008*, Buenos Aires, Argentina, 2008.

[BM58] G. E. P. Box and Mervin E. Müller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, 1958.

[BPW95] C.T.H. Baker, C.A.H. Paul, and D.R. Willé. A Bibliography on the Numerical Solution of Delay Differential Equations. Technical Report Numerical Analysis Report No. 269, University of Manchester, U.K., 1995.

[BS08] J. Brandt and K. Schneider. How different are Esterel and SystemC. *Embedded Systems Specification and Design Languages*, pages 3–13, 2008.

[BTE09] B. Balachandran, Kalmár-Nagy T., and Gilsinn D. Eds. *Delay Differential Equations - Recent Advances and New Directions*. Springer Science+Business Media, 2009.

[Buc00] E. Buckwar. Introduction to the Numerical Analysis of Stochastic Delay Differential Equations. *Journal of Computational and Applied Mathematics*, 125:297–307, 2000.

[But87] J.C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge–Kutta and General Linear Methods*. John Wiley, Chichester, United Kingdom, 1987. 512p.

[BWC10a] Matías Bonaventura, Gabriel A. Wainer, and Rodrigo Castro. Advanced ide for modeling and simulation of discrete event systems. In *Proceedings of the 2010 Spring Simulation Multiconference*, SpringSim '10, pages 125:1–125:8. Society for Computer Simulation International, 2010.

[BWC10b] Matias Bonaventura, Gabriel A. Wainer, and Rodrigo Castro. Advanced Tool for Modeling & Simulation of Discrete Event Systems: A new Eclipse-based Graphical Environment. *Poster Session. XXIV Winter School of Science in Informatics. Univ. of Buenos Aires.*, July 29th, 2010. http://www.dc.uba.ar/inv/PostersECI2010/Poster-ECI2010-Bonaventura-Wainer-Castro.pdf/view.

[BWC12] Matias Bonaventura, Gabriel A. Wainer, and Rodrigo Castro. A Graphical Modeling and Simulation Environment for DEVS. *Simulation*, (2):1–24, October 2012. doi:10.1177/0037549711436267.

[Car03] B. Carlson. *Intel Internet Exchange Architecture and Applications: A Practical Guide to Intel's Network Processors*. Intel Press, 2003.

[Cas93] Christos Cassandras. *Discrete Event Systems: Modeling and Performance Analysis*. Irwin and Aksen, Boston, Massachusetts, 1993.

[CHdV03] Dmitri Chkliaev, Jozef Hooman, and Erik de Vink. Verification and improvement of the sliding window protocol. In *TACAS'03: Proceedings of the 9th international conference on Tools and algorithms for the construction and analysis of systems*, pages 113–127, Berlin, Heidelberg, 2003. Springer-Verlag.

[CK06] F.E. Cellier and E. Kofman. *Continuous System Simulation*. Springer, New York, 2006.

[CK08a] R. Castro and E. Kofman. Discrete Event Modeling and Simulation of Networked Control Systems. Technical Report LSD0808, LSD, UNR, 2008. Available at www.fceia.unr.edu.ar/∼kofman.

[CK08b] R. Castro and E. Kofman. Simulation of NCS systems: A performance study of Discrete Event and Discrete Time techniques. Technical Report LSD0809, LSD, UNR, 2008. Available at www.fceia.unr.edu.ar/∼kofman.

[CKC10a] R. Castro, E. Kofman, and F. Cellier. Quantization based integration methods for delay differential equations. *Simulation Modelling Practice and Theory*, 19(1):314–336, 2010.

[CKC10b] R. Castro, E. Kofman, and F. Cellier. Quantization Based Integration Methods for Delay Differential Equations. Speedups using a Quantization Based Approach. *Poster Session. XXIV Winter School of Science in Informatics. Univ. of Buenos Aires.*, July 29th, 2010. http://www.dc.uba.ar/inv/PostersECI2010/Poster_ECI2010_CastroKofmanCellier_v1.5.jpg/view.

[CKMB08] F. Cellier, E. Kofman, G. Migoni, and M. Bortolotto. Quantized state system simulation. In *Proceedings of SummerSim 08 (2008 Summer Simulation Multiconference)*, Edinburgh, Scotland, 2008.

[CKW08] R. Castro, E. Kofman, and G. Wainer. A formal framework for stochastic DEVS modeling and simulation. In *Proceedings of the 2008 Spring simulation multiconference*, pages 421–428. The Society for Computer Simulation, International, 2008.

[CKW09] R. Castro, E. Kofman, and G. Wainer. A devs-based end-to-end methodology for hybrid control of embedded networking systems. In *Proceedings of ADHS'09: 3rd IFAC Conference on Analysis and Design of Hybrid Systems*, volume 3, pages 74–79, 2009.

[CKW10] R. Castro, E. Kofman, and G. Wainer. A Formal Framework for Stochastic Discrete Event System Specification Modeling and Simulation. *Simulation*, 86(10):587–611, 2010.

[CL04] CG Cassandras and S. Lafortune. *Introduction to discrete event systems*. Kluwer Academic Publishers, 2004.

[Com05] Douglas E. Comer. *Network Systems Design Using Network Processors: Intel 2XXX Version.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2005.

[CRBW12] R. Castro, I. Ramello, M. Bonaventura, and G. Wainer. M&S-based design of embedded controllers on network processors. In *Proceedings of the Symposium on Theory of Modeling and Simulation (TMS/DEVS)*, page 32, Orlando, FL, USA., 2012. Society for Computer Simulation International.

[Cut80] Melvin M. Cutler. Discrete event simulation of stochastic and deterministic sequential machine models. In *Proceedings of the 12th conference on Winter simulation*, pages 83–97, Orlando, FL, 1980.

[EJL+03] J. Eker, J.W. Janneck, E.A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming heterogeneity-the Ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144, 2003.

[FDB02] J.B. Filippi, M. Delhom, and F. Bernardi. The JDEVS Environmental Modeling and Simulation Environment. In *Proceedings of IEMSS 2002*, volume 3, pages 283–288, 2002.

[FH99] S. Floyd and T. Henderson. RFC2582: The NewReno Modification to TCP's Fast Recovery Algorithm. *RFC Editor United States*, 1999.

[FHPW00] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. *ACM SIGCOMM Computer Communication Review*, 30(4):43–56, 2000.

[FJ02] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *Networking, IEEE/ACM Transactions on*, 1(4):397–413, 2002.

[FK03] S. Floyd and E. Kohler. Internet research needs better models. *ACM SIGCOMM Computer Communication Review*, 33(1):29–34, 2003.

[FPEN94] G.F. Franklin, J.D. Powell, and A. Emami Naeini. *Feedback control of dynamic systems.* Addison-Wesley, 1994.

[GD04] R. Gray and L. Davisson. *An Introduction to Statistical Signal Processing.* Cambridge University Press, Cambridge, UK, 2004.

[GEG00] N. Giambiasi, B. Escude, and S. Ghosh. GDEVS: A generalized Discrete Event specification for accurate modeling of dynamic systems. *Transactions of SCS*, 17(3):120–134, 2000.

[GH01] G.C.Walsh and H.Ye. Scheduling of networked control systems. *IEEE Control Systems Magazine*, 21(1):57–65, 2001.

[GKS04] A. Gavrilovska, S. Kumar, and K. Schwan. The execution of event-action rules on programmable network processors. In *Workshop on Operating System and Architectural Support for the On-Demand IT Infrastructure (OASIS 2004), held with ASPLOS-XI*, 2004.

[GLT04] Yu Gu, Yong Liu, and Don Towsley. On integrating fluid models with packet simulation. In *In Proceedings of IEEE INFOCOM*, 2004.

[GM02] Luigi Alfredo Grieco and Saverio Mascolo. Tcp westwood and easy red to improve fairness in high-speed networks. In *PIHSN '02: Proceedings of the 7th IFIP/IEEE International Workshop on Protocols for High Speed Networks*, pages 130–146, London, UK, 2002. Springer-Verlag.

[GM04] Luigi A. Grieco and Saverio Mascolo. Performance evaluation and comparison of westwood+, new reno, and vegas tcp congestion control. *SIGCOMM Comput. Commun. Rev.*, 34(2):25–38, 2004.

[GSH88] A. Greiner, W. Strittmatter, and J. Honerkamp. Numerical Integration of Stochastic Differential Equations. *Journal of Statistical Physics*, 51(1-2):95–108, 1988.

[Hel04] J. Hellerstein. *Feedback control of computing systems.* Wiley-IEEE Press, 2004.

[Hig01] D. J. Higham. An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations. *SIAM Review*, 43(3):525–546, 2001.

[HLW02] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration Structure-Preserving Algorithms for Ordinary Differential Equations.* Springer, 2002.

[HMTG01] CV Hollot, V. Misra, D. Towsley, and W.B. Gong. On designing improved controllers for AQM routers supporting TCP flows. In *IEEE INFOCOM 2001*, volume 3, 2001.

[HMY07] D. J. Higham, Xuerong Mao, and Chenggui Yuan. Preserving exponential mean-square stability in the simulation of hybrid stochastic differential equations. *Numerische Mathematik*, 108(2):295–325, 2007.

[HNW00] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8 of *Series in Computational Mathematics.* Springer–Verlag, Berlin, 2^{nd} edition, 2000. 528p.

[HNX07] J.P. Hespanha, P. Naghshtabrizi, and Y. Xu. A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1):138–162, 2007.

[HS04] D. Huang and H. Sarjoughian. Software and simulation modeling for real-time software-intensive systems. In *Distributed Simulation and Real-Time Applications, 2004. DS-RT 2004. Eighth IEEE International Symposium on*, pages 196–203. IEEE, 2004.

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.* Springer, Berlin, 1991.

[IH08] Teerawat Issariyakul and Ekram Hossain. *Introduction to Network Simulator NS2.* Springer, 2008.

[Ins09] Institute for Statistics of the Vienna University of Economics and Business Administration. ARVAG Project - Automatic nonuniform Random VAriate Generation. http://statistik.wu-wien.ac.at/arvag/, 2009.

[Int04] Intel. *Intel IXP2400 Network Processors.* Intel Press, 2004.

[Jai89] R. Jain. A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks. *ACM SIGCOMM Computer Communication Review*, 19(5):56–71, 1989.

[JK88] Van Jacobson and Michael J. Karels. Congestion avoidance and control. In *Proceedings of SIGCOMM88.* ACM, August 1988.

[Jos96] C. Joslyn. The Process Theoretical Approach to Qualitative DEVS. In *Proc. 7th Conf. on AI, Simulation, and Planning in High Autonomy Systems (AIS '96)*, pages 235–242, San Diego, California, 1996.

[JWBC09] Shafagh Jafer, Gabriel A. Wainer, Matias Bonaventura, and Rodrigo Castro. Conservative algorithms for flattened devs and cell-devs simulators (in spanish). In *Proceedings of HPC LatinAmerica 2009*, Mar del Plata, Argentina, 2009.

[KC06] E. Kofman and R.D. Castro. STDEVS, A Novel Formalism for Modeling and Simulation of Stochastic Discrete Event Systems. In *Proceedings of AADECA 2006*, Buenos Aires, Argentina, 2006.

[Kel01] F. Kelly. Mathematical modeling of the internet. In B. Engquist and W. Schmid, editors, *Mathematics Unlimited-2001 and Beyond*, pages 685–702. Springer-Verlag, Berlin, 2001.

[KJ01] E. Kofman and S. Junco. Quantized state systems. a devs approach for continuous system simulation. *Transactions of SCS*, 18(3):123–132, 2001.

[KK00] W.H. Kwon and H.S. Kim. A survey of control theoretic approaches in wired and wireless communication networks. In *Proceedings of the Korea-Japan Joint Workshop*, pages 30–45, 2000.

[Kle75] Leonard Kleinrock. *Queuing Systems, Vol.1: Theory.* Wiley & Sons, New York, NY, USA, 1975.

[KLP03] E. Kofman, M. Lapadula, and E. Pagliero. PowerDEVS: A DEVS Based Environment for Hybrid System Modeling and Simulation. Technical Report LSD0306, LSD, UNR, 2003.

[KLP04] Valeri Kontorovitch, Vladimir Lyandres, and Serguei Primak. *Stochastic Methods and their Applications to Communications: Stochastic Differential Equations Approach.* John Wiley & Sons, Inc., New York, NY, USA, 2004.

[KM01] Krishnan Kumaran and Debasis Mitra. Performance and fluid simulations of a novel shared buffer management system. *ACM Trans. Model. Comput. Simul.*, 11(1):43–75, 2001.

[Knu97] Donald E. Knuth. *Art of Computer Programming, Volume 2: Seminumerical Algorithms (3rd Edition).* Addison-Wesley Professional, November 1997.

[Kof02] E. Kofman. A Second Order Approximation for DEVS Simulation of Continuous Systems. *Simulation: Transactions of the Society for Modeling and Simulation International*, 78(2):76–89, 2002.

[Kof04] E. Kofman. Discrete Event Simulation of Hybrid Systems. *SIAM Journal on Scientific Computing*, 25(5):1771–1797, 2004.

[Kof06] E. Kofman. A Third Order Discrete Event Simulation Method for Continuous System Simulation. *Latin American Applied Research*, 36(2):101–108, 2006.

[KRW03] M. Kihl, A. Robertsson, and B. Wittenmark. Analysis of admission control mechanisms using non-linear control theory. In *Proceedings of ISCC 2003*, volume 2, pages 1306–1311, Kiris-Kemer, Turkey, 2003.

[Lam91] J.D. Lambert. *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem.* John Wiley & Sons, 1991.

[LBS08] S. Liu, T. Basar, and R. Srikant. TCP-Illinois: A loss-and delay-based congestion control algorithm for high-speed networks. *Performance Evaluation*, 65(6-7):417–440, 2008.

[LdF08] M.A.E. Lima and N.L.S. da Fonseca. Active Queue Management. *Encyclopedia of Internet technologies and applications*, page 1, 2008.

[Lee04] B. Lee. Eclipse Project CDT (C/C++) Plugin Tutorial. *Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada February*, 20, 2004.

[Lit61] J.D.C. Little. A Proof for the Queuing Formula: L= λ W. *Operations Research*, 9(3):383–387, 1961.

[Mas05] Saverio Mascolo. Modeling and designing the internet congestion control. In Sophie Tarbouriech, Chaouki T. Abdallah, and John Chiasson, editors, *Advances in Communication Control Networks*, chapter 7, pages 33–36. Springer, 2005.

[Mel76] B. Melamed. *Analysis and Simplifications of Discrete Event Systems and Jackson Queuing Networks.* PhD thesis, The University of Michigan,, Ann Arbor, Michigan, 1976.

[MGT99] V. Misra, W. Gong, and D. Towsley. Stochastic differential equation modeling and analysis of TCP-windowsize behavior. In *Proc. Performance'99*, 1999.

[MGT00] V. Misra, W. Gong, and D. Towsley. A Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. In *Proc. SIGCOMM 2000*, pages 151–160. ACM, 2000.

[Mig10] G. Migoni. *Simulación por Cuantificación de Sistemas Stiff.* PhD thesis, Facultad de Ciencias Exactas, Ingeniería y Agrimensura. Universidad Nacional de Rosario, Rosario, Argentina, 2010.

[MK09] G. Migoni and E. Kofman. Linearly Implicit Discrete Event Methods for Stiff ODEs. *Latin American Applied Research*, 39(3):245–254, 2009.

[MN05] W. Michiels and S.I. Niculescu. Stability analysis of a fluid flow model for TCP like behavior. *International Journal of Bifurcation and Chaos*, 15(7):2277–2282, 2005.

[MS90] D. Mitra and J.B. Seery. Dynamic adaptive windows for high speed data networks: theory and simulations. *ACM SIGCOMM Computer Communication Review*, 20(4):30–40, 1990.

[Nag84] J. Nagle. Congestion control in IP/TCP internetworks. *ACM SIGCOMM Computer Communication Review*, 14(4):17, 1984.

[NBW98] J. Nilsson, B. Bernhardsson, and B. Wittenmark. Stochastic analysis and control of real-time systems with random time delays. *Automatica*, 34(1):57–64, 1998.

[Nut03] James Nutaro. *Parallel Discrete Event Simulation with Application to Continuous Systems.* PhD thesis, The University of Arizona, 2003.

[OMN04] OMNeT++ Community. OMNeT++ Discrete Event Simulation System. www.omnetpp.org, 2004.

[OP81] H.J. Oberle and H.J. Pesch. Numerical Treatment of Delay Differential Equations by Hermite Interpolation. *Numerische Mathematik*, 37(2):235–255, 1981.

[PFTK98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling tcp throughpu: A simple model and its empirical validation. Technical report, University of Massachusetts, Amherst, MA, USA, 1998.

[PFTK00] J. Padhye, V. Firoiu, D.F. Towsley, and J.F. Kurose. Modeling TCP Reno performance: a simple model and its empirical validation. *IEEE/ACM Transactions on Networking (ToN)*, 8(2):133–145, 2000.

[PJ06] P. Pepe and Z.-P. Jiang. A Lyapunov−Krasovskii methodology for ISS and $iISS$ of time-delay systems. *Systems Control Letters.*, 55(12):1006–1014, December 2006.

[PTVF07] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, August 2007.

[PW00] Kihong Park and Walter Willinger. *Self-Similar Network Traffic and Performance Evaluation.* John Wiley & Sons, Inc., New York, NY, USA, 2000.

[Rad06] Radisys. ENP-2611 Data Sheet. http://www.radisys.com/products/datasheets/ENP-2611.pdf, 2006.

[Rüm82] W. Rümelin. Numerical Treatment of Stochastic Differential Equations. *SIAM Journal on Numerical Analysis*, 19(3):604–613, 1982.

[SA91] D. Sanghi and A.K. Agrawal. DTP: An efficient transport protocol. In *In Proceedings of the IFIP TC6 Working Conference on ComACM Computing Surveys*, 1991.

[SLCP01] J. Schormans, E. Liu, L. Cuthbert, and J. Pitts. A hybrid technique for accelerated simulation of ATM networks and network elements. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 11(2):182–205, 2001.

[SM96] Y. Saito and T. Mitsui. Stability Analysis of Numerical Schemes for Stochastic Differential Equations. *SIAM Journal on Numerical Analysis*, 33(6):2254–2267, 1996.

[S.P08] S.P. Corwin, S. Thompson and S.M. White. Solving ODEs and DDEs with Impulses. *JNAIAM J. Numer. Anal. Indust. Appl.*, 3:139–149, 2008.

[ST00] L.F. Shampine and S. Thompson. Solving Delay Differential Equations with dde23. Technical report, Southern Methodist University, 2000.

[ST01] L.F. Shampine and S. Thompson. Solving DDEs in Matlab. *Applied Numerical Mathematics*, 37(4):441 – 458, 2001.

[Sta96] J.A. Stankovic. Strategic directions in real-time and embedded systems. *ACM Computing Surveys (CSUR)*, 28(4):751–763, 1996.

[SW94] W.R. Stevens and G.R. Wright. *TCP/IP Illustrated: The protocols*, volume 1. Addison-Wesley Professional, 1994.

[SW95] W.R. Stevens and G.R. Wright. *TCP/IP Illustrated: The Implementation*, volume 2. Addison-Wesley Professional, 1995.

[Vit05] Giuseppe Vitali. Sul problema della mesura dei gruppi di punti di una retta. Bologna, Italy, 1905.

[Wai02] G. Wainer. CD++: a toolkit to define discrete-event models. *Software, Practice and Experience*, 32(13):1261–1306, 2002.

[Wai09] G.A. Wainer. *Discrete-Event Modeling and Simulation: A Practitioner's Approach*. CRC Press (in print), 2009.

[WB92] D.R. Willé and C.T.H. Baker. DELSOL – A Numerical Code for the Solution of Systems of Delay–Differential Equations. *Applied Numerical Mathematics*, 9(3–5):223–234, 1992.

[WC91] Z. Wang and J. Crowcroft. A new congestion control scheme: Slow start and search (Tri-S). *ACM SIGCOMM Computer Communication Review*, 21(1):32–43, 1991.

[WC92] Z. Wang and J. Crowcroft. Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm. *ACM SIGCOMM Computer Communication Review*, 22(2):9–16, 1992.

[WC09] Gabriel A. Wainer and Rodrigo Castro. A survey on the application of the cell-devs formalism in cellular models. *Journal of Cellular Automata*, 2009. accepted: October 2008.

[WC11] G. Wainer and R. Castro. DEMES: a Discrete-Event Methodology for Modeling and Simulation of Embedded Systems. *SCS Modeling & Simulation Magazine*, 2:65–73, 2011.

[WCD01] G. Wainer, G. Christen, and A. Dobniewski. Defining DEVS Models with the CD++ Toolkit. In *Proceedings of ESS2001*, pages 633–637, Marseille, France, 2001. SCS Publisher.

[WGM05] Gabriel A. Wainer, Ezequiel Glinsky, and Peter MacSween. A model-driven technique for development of embedded systems based on the devs formalism, 2005.

[YC07] Y. Yongqing and J. Cao. Stability and Periodicity in Delayed Cellular Neural Networks with Impulsive Effects. *Nonlinear Anal. Real World Appl.*, 8:362–374, 2007.

[YL00] Y.R. Yang and S.S. Lam. General AIMD congestion control. In *icnp*, page 187. Published by the IEEE Computer Society, 2000.

[YS07] Yung Yi and Sanjay Shakkottai. Flunet: A hybrid internet simulator for fast queue regimes. *Comput. Netw.*, 51(18):4919–4937, 2007.

[YW07] Y.H. Yu and G. Wainer. ECD++: an engine for executing DEVS models in embedded platforms. In *Proceedings of the 2007 summer computer simulation conference*, pages 323–330. Society for Computer Simulation International San Diego, CA, USA, 2007.

[Zei76] B. Zeigler. *Theory of Modeling and Simulation*. John Wiley & Sons, New York, 1976.

[Zim02] H. Zimmermann. OSI reference model–The ISO model of architecture for open systems interconnection. *Communications, IEEE Transactions on*, 28(4):425–432, 2002.

[ZKP00] B. Zeigler, T.G. Kim, and H. Praehofer. *Theory of Modeling and Simulation. 2nd. edition.* Academic Press, New York, 2000.

[ZS00] Bernard Zeigler and Hessam Sarjoughian. *Introduction to DEVS Modeling and Simulation with JAVA: A Simplified Approach to HLA-Compliant Distributed Simulations.* Arizona Center for Integrative Modeling and Simulation, 2000.

[ZSL09] Guihua Zhao, Minghui Sing, and Mingzhu Lio. Numerical Solutions of Stochastic Differential Delay Equations with Jumps. *International Journal of Numerical Analysis and Modeling*, 6(4):659–679, 2009.

Apéndice A

Pruebas de Teoremas

A.1. Estabilidad Entrada–Estado

Se definen aquí algunas herramientas que se utilizarán en los teoremas y sus pruebas.

Función Clase $\mathcal{K}$ Una función continua a valores reales $\alpha : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ se dice pertenecer a la clase $\mathcal{K}$ si la misma es estrictamente creciente y $\alpha(0) = 0$.

Función Clase $\mathcal{L}$ Una función continua a valores reales $\psi : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ se dice pertenecer a la clase $\mathcal{L}$ si la misma es estrictamente decreciente y $\lim_{t\to\infty} \psi(t) = 0$.

Función Clase $\mathcal{KL}$ Una función continua a valores reales $\beta : \mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{R}_0^+$ se dice pertenecer a la clase $\mathcal{KL}$ si la misma es Clase $\mathcal{K}$ con respecto al primer argumento y Clase $\mathcal{L}$ con respecto al segundo.

Estable Entrada–Estado (ISS, por *Input to State Stable*) Sea $\phi_a(t)$ la solución de la DDE

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{x}(t - \tau_1(\mathbf{x}, t)), ..., \mathbf{x}(t - \tau_m(\mathbf{x}, t)), \mathbf{u}(t)) \tag{A.1}$$

para $\mathbf{u}(t) = 0$ con una historia inicial dada por $\phi_a(t < 0)$, y sea $\phi(t)$ la solución para una entrada dada por $\mathbf{u}(t)$ con historia inicial $\phi(t < 0)$.

Se define

$$\Delta\phi_0 = \sup_{t<0} \|\phi(t) - \phi_a(t)\| \tag{A.2}$$

y

$$u_m = \sup_{t\geq 0} \|\mathbf{u}(t)\| \tag{A.3}$$

Se dice que el sistema de la Ec.(A.1) es ISS sobre $\phi_a(t)$ si existe una función β clase $\mathcal{KL}$ y una función α clase $\mathcal{K}$ tal que

$$\|\phi(t) - \phi_a(t)\| \leq \beta(\Delta\phi_0, t) + \alpha(u_m) \tag{A.4}$$

Cuando la historia inicial de $\phi(t)$ es idéntica a la de $\phi_a(t)$, se tiene $\Delta\phi_0 = 0$ y la condición ISS deviene en:

$$\|\phi(t) - \phi_a(t)\| \leq \alpha(u_m) \tag{A.5}$$

En este último caso, la propiedad de ser ISS implica que una entrada acotada provocará una diferencia acotada entre las trayectorias del sistema forzado y del sistema libre. Adicionalmente, a medida que se hace más pequeña su cota superior, también lo hace la cota superior de la diferencia entre ambas trayectorias.

A.2. Prueba del Teorema 1

La aproximación QSS de la Ec. (4.16) está dada por:

$$\dot{\mathbf{x}}(t) = A\mathbf{q}(t) + \sum_{i=1}^{m} A_i \mathbf{q}(t - \tau_i) \tag{A.6}$$

Sea $\phi_a(t)$ la solución analítica de la Ec.(4.16) partiendo de una historia inicial arbitraria, y sea $\phi(t)$ la solución de la Ec. (A.6) partiendo de la misma condición inicial $\phi(t \leq 0) = \phi_a(t \leq 0)$.

Defínase la perturbación introducida en los estados mediante la cuantización de la Ec. (A.6) como:

$$\boldsymbol{\Delta}\mathbf{x}(\mathbf{t}) \triangleq \mathbf{q}(t) - \mathbf{x}(t) \tag{A.7}$$

Ahora, se reescribe la Ec. (A.6) como:

$$\dot{\mathbf{x}}(t) = A(\mathbf{x}(t) + \boldsymbol{\Delta}\mathbf{x}(t)) + \sum_{i=1}^{m} A_i(\mathbf{x}(t - \tau_i) + \boldsymbol{\Delta}\mathbf{x}(t - \tau_i)) \tag{A.8}$$

Esto implica que $\phi(t)$ será también la solución de la Ec. (A.8), verificándose:

$$\dot{\phi}(t) = A(\phi(t) + \boldsymbol{\Delta}\mathbf{x}(t)) + \sum_{i=1}^{m} A_i(\phi(t - \tau_i) + \boldsymbol{\Delta}\mathbf{x}(t - \tau_i)) \tag{A.9}$$

Como $\phi_a(t)$ es una solución de la Ec. (4.16), lo siguiente se verificará:

$$\dot{\phi}_a(t) = A\phi_a(t) + \sum_{i=1}^{m} A_i \phi_a(t - \tau_i) \tag{A.10}$$

Defínase ahora el error cometido por la aproximación QSS como:

$$\mathbf{e}(t) \triangleq \phi(t) - \phi_a(t) \tag{A.11}$$

Luego, sustrayendo la Ec.(A.10) de la Ec.(A.9) el error tendrá la siguiente dinámica:

$$\dot{\mathbf{e}}(\mathbf{t}) = A\mathbf{e}(t) + \sum_{i=1}^{m} A_i \mathbf{e}(t - \tau_i) + A\boldsymbol{\Delta}\mathbf{x}(t) + \sum_{i=1}^{m} A_i \boldsymbol{\Delta}\mathbf{x}(t - \tau_i) \tag{A.12}$$

con $\mathbf{e}(t \leq 0) = 0$.

Definiendo $B \triangleq I$ y

$$\mathbf{u}(t) \triangleq A\boldsymbol{\Delta}\mathbf{x}(t) + \sum_{i=1}^{m} A_i \boldsymbol{\Delta}\mathbf{x}(t-\tau_i) \tag{A.13}$$

la dinámica del error puede reescribirse como:

$$\dot{\mathbf{e}}(t) = A\mathbf{e}(t) + \sum_{i=1}^{m} A_i \mathbf{e}(t-\tau_i) + B\mathbf{u}(t) \tag{A.14}$$

Cuando $\mathbf{u}(t) = 0$, esta última ecuación es idéntica a la Ec. (4.16) con la condición inicial trivial. Luego, de acuerdo a la hipótesis inicial, la solución de (A.14) para $\mathbf{u}(t) = 0$ es asintóticamente estable. Siguiendo la Prop. 2.5 de [PJ06], esta condición implica que la Ec. (A.14) es Estable Entrada–Estado (ISS).

Esto es, existe una función α que es Clase $\mathcal{K}$ que verifica $\|\mathbf{e}(t)\| \leq \alpha(\sup_{t\geq 0}(\|\mathbf{u}(t)\|))$. Luego, teniendo en cuenta que

$$\begin{aligned}
\|\mathbf{u}(t)\| &= \|A\boldsymbol{\Delta}\mathbf{x}(t) + \sum_{i=1}^{m} A_i \boldsymbol{\Delta}\mathbf{x}(t-\tau_i)\| \\
&\leq \|A\| \cdot \|\boldsymbol{\Delta}\mathbf{x}(t)\| + \sum_{i=1}^{m} \|A_i\| \cdot \|\boldsymbol{\Delta}\mathbf{x}(t-\tau_i)\| \\
&\leq \|A\| \cdot \|\boldsymbol{\Delta}\mathbf{Q}\| + \sum_{i=1}^{m} \|A_i\| \cdot \|\boldsymbol{\Delta}\mathbf{Q}\| \\
&\leq (\|A\| + \sum_{i=1}^{m} \|A_i\|) \cdot \|\boldsymbol{\Delta}\mathbf{Q}\|
\end{aligned} \tag{A.15}$$

siendo $\boldsymbol{\Delta}\mathbf{Q}$ el vector de quantum, resulta que

$$\|\mathbf{e}\| \leq \alpha((\|A\| + \sum_{i=1}^{m} \|A_i\|) \cdot \|\boldsymbol{\Delta}\mathbf{Q}\|) \triangleq \gamma(\|\boldsymbol{\Delta}\mathbf{Q}\|) \tag{A.16}$$

donde γ es una función Clase $\mathcal{K}$.

Luego, el error es acotado para todo $t \geq 0$. Esto completa la prueba del ítem 1).

Aún mas, cuando el quantum tiende a cero, es decir, $\|\boldsymbol{\Delta}\mathbf{Q}\| \to 0$, resulta que $\|\mathbf{e}\| \to 0$ (esto es una propiedad de cualquier función Clase $\mathcal{K}$). Esto prueba la condición de convergencia postulada en el ítem 2). ■

A.3. Prueba del Teorema 2

Sea e_m un numero positivo arbitracior. Sea $t_f > 0$ el primer instante en el cual la norma del error $\|\mathbf{e}(t)\|$ alcanza el valor e_m. Luego, se tiene $\|\mathbf{e}(t)\| < e_m$ para todo $t < t_f$.

Defínase

$$\boldsymbol{\Delta}\mathbf{x}(t) \triangleq \mathbf{q}(t) - \mathbf{x}(t) \tag{A.17}$$

y

$$\boldsymbol{\Delta}\mathbf{x}_\tau(t) \triangleq \mathbf{x}(t - \tau_1(\mathbf{q}, t)) - \mathbf{x}(t - \tau_1(\mathbf{x}, t)) \tag{A.18}$$

Luego, puede reescribirse la aproximación DQSS de la Ec.(4.19) como

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t) + \mathbf{\Delta x}(t), \mathbf{x}(t - \tau_1(\mathbf{x}, t)) + \mathbf{\Delta x}_\tau(t) + \mathbf{\Delta x}(t - \tau_1(\mathbf{q}, t)), t) \quad \text{(A.19)}$$

Defínase ahora

$$\begin{aligned}\mathbf{u}(t) \triangleq &\mathbf{f}(\mathbf{x}(t) + \mathbf{\Delta x}(t), \mathbf{x}(t - \tau_1(\mathbf{x}, t)) + \mathbf{\Delta x}_\tau(t) + \mathbf{\Delta x}(t - \tau_1(\mathbf{q}, t)), t) \\ &- \mathbf{f}(\mathbf{x}(t), \mathbf{x}(t - \tau_1(\mathbf{x}, t)), t)\end{aligned} \quad \text{(A.20)}$$

Luego, la aproximación DQSS de la Ec.(A.19) se hace idéntica al sistema de la Ec.(4.18), que es ISS sobre la solución analítica $\phi_a(t)$.

También se sabe que

$$\|\mathbf{\Delta x}(t)\| \leq \delta Q \triangleq \|\Delta Q\| \quad \text{(A.21)}$$

para todo t. Notar asimismo que, dado que se asume que $\|\phi(t) - \phi_a(t)\| < e_m$ para todo $t < t_f$, luego la función $\mathbf{f}$ en la aproximación DQSS de la Ec.(4.19) está siendo evaluada en una región acotada, ya que

$$\|\mathbf{q}(t)\| \leq \phi_{a_{max}} + e_m + \delta Q \quad \text{(A.22)}$$

para todo $t < t_f$, siendo $\phi_{a_{max}}$ una cota superior para la norma de la solución analítica $\phi_a(t)$.

Teniendo en cuenta que $\mathbf{f}$ es localmente Lipschitz, existe una constante L para la región dada tal que

$$\|\mathbf{u}(t)\| \leq L \cdot (\|\mathbf{\Delta x}(t)\| + \|\mathbf{\Delta x}_\tau(t) + \mathbf{\Delta x}(t - \tau_1(\mathbf{q}, t))\|) \quad \text{(A.23)}$$

Usando también la Ec.(A.21) en esta última ecuación resulta que

$$\|\mathbf{u}(t)\| \leq L \cdot (2\delta Q + \|\mathbf{\Delta x}_\tau(t)\|) \quad \text{(A.24)}$$

La función $\tau_1(\cdot)$ es también localmente Lipschitz y está siendo evaluada en la misma región acotada. Luego, existe una constante L_τ en esa región tal que:

$$|\tau_1(\mathbf{q}(t), t) - \tau_1(\mathbf{x}(t), t)| \leq L_\tau \cdot \|\mathbf{q}(t) - \mathbf{x}(t)\| \leq L_\tau \cdot \delta Q \quad \text{(A.25)}$$

Partiendo de la Ec.(A.18) se sabe que

$$\|\mathbf{\Delta x}_\tau(t)\| \leq \max_{t<t_f}(\|\dot{\mathbf{x}}(t)\|) \cdot |\tau_1(\mathbf{q}(t), t)) - \tau_1(\mathbf{x}(t), t))| \quad \text{(A.26)}$$

Recordando que $\mathbf{f}(\cdot)$ está siendo evaluada en una región acotada definida por la Ec.(A.22) y que es localmente Lipschitz, resulta también acotada, y luego existe una constante positiva f_m tal que $(\|\dot{\mathbf{x}}(t)\|) \leq f_m$ para $t < t_f$. Luego,

$$\|\mathbf{\Delta x}_\tau(t)\| \leq f_m \cdot L_\tau \cdot \delta Q \quad \text{(A.27)}$$

Insertando esta última desigualdad en la Ec.(A.24), se obtiene:

$$\|\mathbf{u}(t)\| \leq (2 \cdot L + f_m \cdot L_\tau) \cdot \delta Q \quad \text{(A.28)}$$

Dado que el sistema forzado de la Ec.(4.18) es ISS y $\phi(t) - \phi_a(t) = 0$ para $t \leq 0$, luego la última ecuación implica que existe una función α Clase $\mathcal{K}$ tal que

$$\|\mathbf{e}(t)\| \leq \alpha((2 \cdot L + f_m \cdot L_\tau) \cdot \delta Q) \quad \text{(A.29)}$$

para todo $t < t_f$.

Luego, tomando un quanta lo suficientemente pequeño δQ, puede hacerse

$$\|\mathbf{e}(t)\| \leq \alpha((2 \cdot L + f_m \cdot L_\tau) \cdot \delta Q) < \rho \cdot e_m \tag{A.30}$$

para cualquier $0 < \rho < 1$.

Esto contradice la suposición inicial de que el error alcanza al valor e_m en el instante t_f. Así, eligiendo el quantum de modo que se cumpla la Ec.(A.30), el error nunca alcanzará e_m.

Aún mas, se se toma un valor arbitrariamente pequeño para e_m, siempre puede encontrarse un δQ tal que se verifica la Ec.(A.30). Esto prueba la convergencia cuando $\delta Q \to 0$. ■

Apéndice B

Biblioteca de Modelos de Red

B.1. Modelos de Red

Una necesidad relevante es la de simplificar al máximo posible la tarea práctica de modelar y simular redes y sus controles utilizando una herramienta basada en DEVS. Si bien los nuevos aportes de esta Tesis con respecto a la representación de fenómenos estocásticos (mediante STDEVS, ver Capítulo 3) solucionan limitaciones teóricas preexistentes, sería muy engorroso para la comunidad de usuarios tener que desarrollar de cero los modelos de red que implementen la nueva teoría, repitiendo el esfuerzo ante cada nuevo sistema.

A lo largo de los capítulos de aplicaciones 6 y 7 se desarrollaron y aplicaron en PowerDEVS modelos DEVS atómicos para representar elementos de red típicos. Estos componen una biblioteca de modelos reutilizables para diseñar redes y controladores generalizados. A continuación se listan dichos modelos acompañados por una breve referencia a su funcionamiento e interfaz de usuario.

Los nuevos modelos incluyen funcionalidades de bajo nivel para manipular estructuras de paquetes de datos tan complejas como se necesite; siguiendo un enfoque de estratificación por capas de los protocolos.

B.1.1. Generador de Paquetes

- **Descripción Funcional.**

  Crea y emite paquetes de red. Los paquetes contienen los datos a transportar ($Payload$) y una estructura multiprotocolo y jerárquica de tipo $E_{ID} = \langle Protocol_{ID}, Header_{ID}, Payload_{ID}, Trailer_{ID} \rangle$ en donde cada $Payload_{ID}$ puede a su vez componerse por una nueva estructura según $Payload_{ID} = E_{ID+1}$ recursivamente.

  En la capa más profunda $Payload$ no contiene información de protocolos sino solo los datos originales que se pretendieron enviar desde la capa más alta (capa "de aplicación", o "de usuario").

  El flujo de paquetes puede ser asignado con un Identificador de Flujo ($FlowID$) para su posterior enrutamiento y procesamiento.

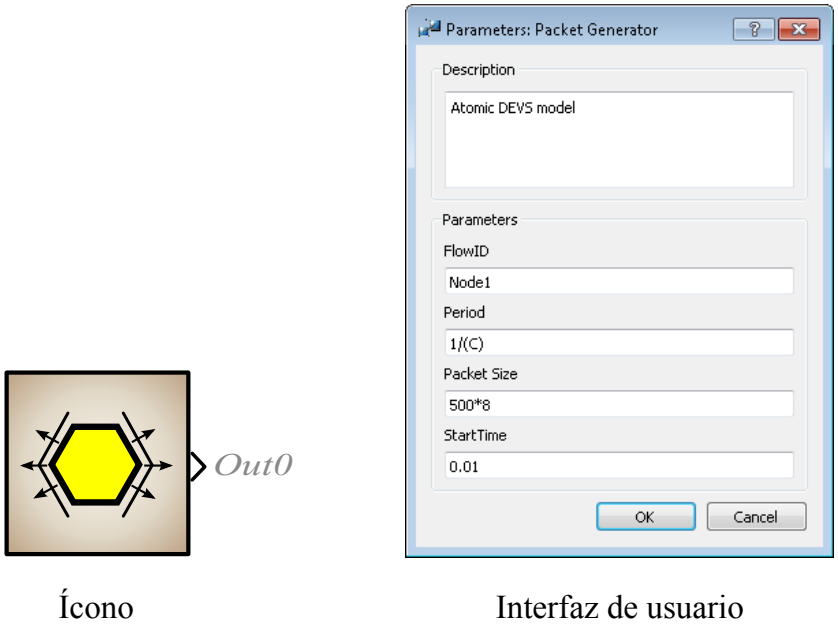


Figura B.1: PacketGenerator.

- Puertos de Entrada.

No tiene.

- Puertos de Salida.

Out0 Puerto por donde se emiten los paquetes.

- Interfaz de Usuario.

FlowID Identificación de Flujo común a todos los paquetes. A utilizar en otros elementos de red para propósito de enrutamiento, gestión de calidad de tráfico, consumo de la información, etc.

Period Tiempo promedio de intergeneración de cada paquete (en segundos). Si se especifica una distribución de probabilidades, este valor funciona como el valor medio de la distribución.

PacketSize Tamaño del paquete (en bits).

StartTime Tiempo de demora (inactividad) hasta que el bloque comienza a emitir paquetes (en segundos).

B.1.2. Destino de Paquetes

- **Descripción Funcional.**

  Recibe y destruye paquetes. Solo procesa los paquetes pertenecientes a FlowID. El procesamiento se define internamente. Por defecto no se aplica ningún procesamiento y simplemente se destruye el paquete recibido.

- Puertos de Entrada.

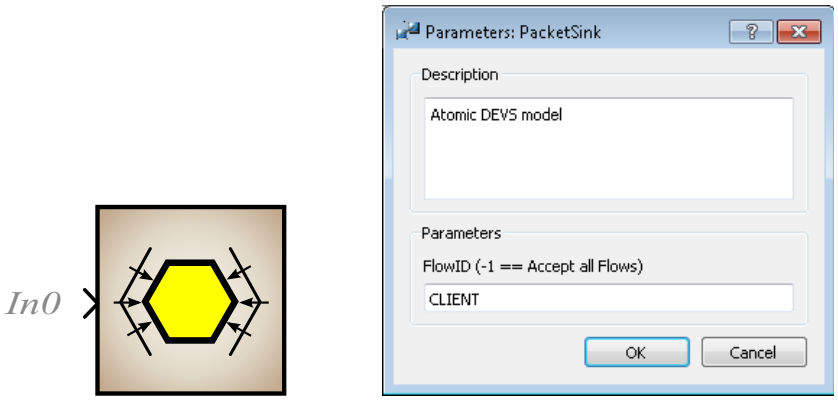


Figura B.2: PacketSink.

In0 Puerto por donde se reciben los paquetes.

- Puertos de Salida.

Out0 No tiene.

- Interfaz de Usuario.

FlowID Identificador de flujo según el cual se filtran los paquetes entrantes. (String. ¨1ädesactiva el filtro).

B.1.3. Multiplexor

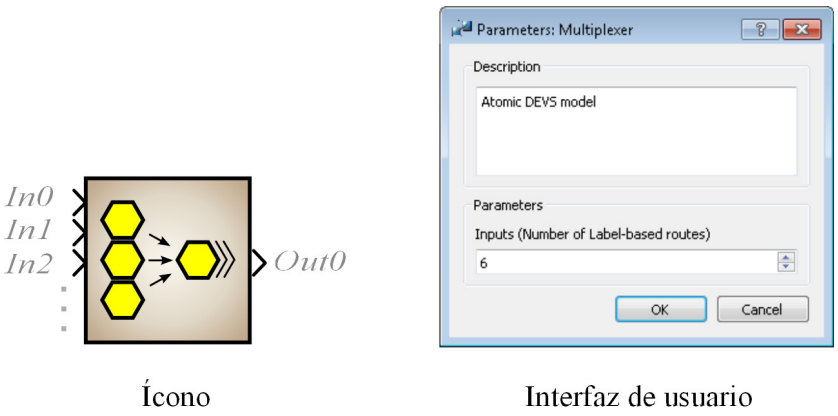


Figura B.3: Multiplexer.

- **Descripción Funcional.**

  Multiplexa N flujos de paquetes en un único flujo de salida, respetando la secuencia temporal de ingreso de cada paquete.

- Puertos de Entrada.

In0...In(N-1) Puertos de ingreso de cada flujo (1 a N) de paquetes.

- Puertos de Salida.

Out0 Puerto de salida del nuevo flujo multiplexado.

- Interfaz de Usuario.

Inputs Cantidad de flujos de entrada.

B.1.4. DeMultiplexor

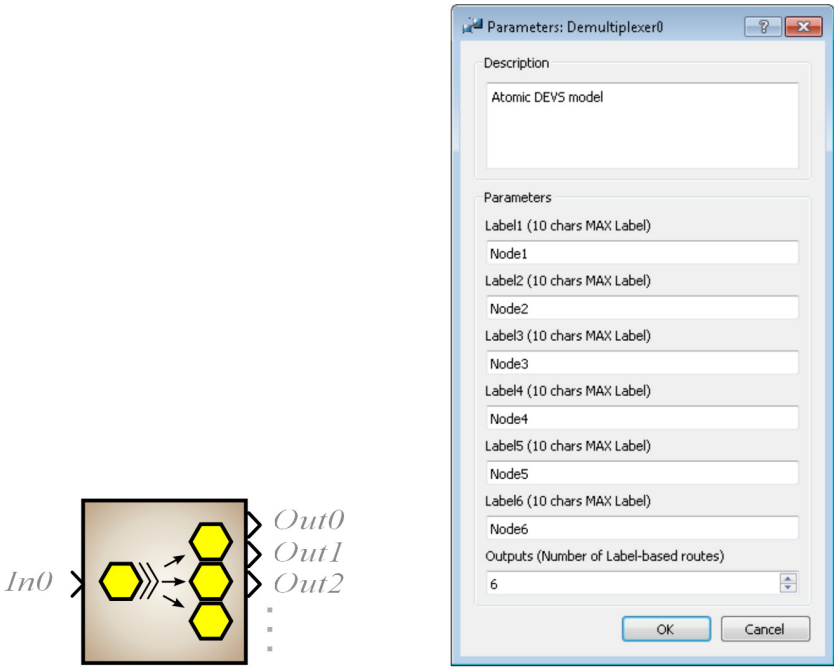


Figura B.4: DeMultiplexer.

- **Descripción Funcional.**

  Demultiplexa un único flujo de paquetes de entrada en N flujos de salida. Cada flujo de salida contiene paquetes pertenecientes a un único identificador de flujo.

- Puertos de Entrada.

  In0 Puerto de ingreso del flujo de paquetes a demultiplexar.

- Puertos de Salida.

Out0...Out(N-1) Puertos de salida de cada nuevo flujo demultiplexado.

- Interfaz de Usuario.

Label1...LabelN Identificadores de flujo para filtrar paquetes en cada puerto de salida.

Outports Cantidad de flujos de salida.

B.1.5. Cola

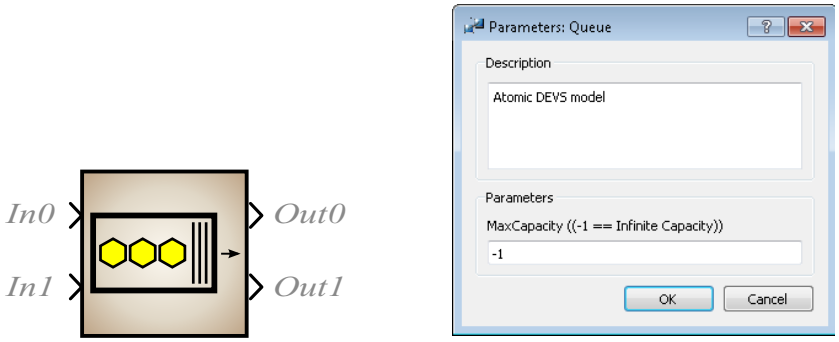


Figura B.5: Queue.

- **Descripción Funcional.**

  Encola paquetes de entrada y los desencola según sean requeridos por una entidad externa. Política FIFO (FirstInFirstOut). Provee información de su estado interno.

- Puertos de Entrada.

  In0 Puerto de ingreso del flujo de paquetes a encolar.

  In1 Puerto de ingreso de la señal de pedido del próximo paquete.

- Puertos de Salida.

  Out0 Puerto de salida de los paquetes desencolados.

  Out1 Puerto de salida de la información de la longitud actual de la cola. Actualiza la información ante cada entrada y salida de paquetes.

- Interfaz de Usuario.

MaxCapacity Determina la cantidad máxima de paquetes que puede encolar. -1 indica capacidad infinita. (en paquetes).

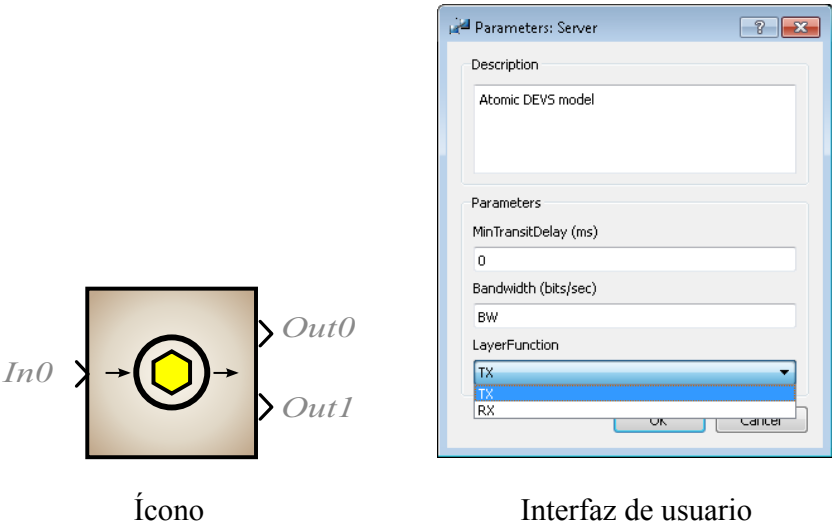


Figura B.6: Server.

B.1.6. Servidor

- **Descripción Funcional.**

  Procesa paquetes imponiendo tiempos de retardo de servicio. Manipula capas de protocolos de los paquetes para imitar un servicio en un capa OSI determinada. Puede agregar una estructura de protocolo (función de Transmisor: toma un paquete proveniente de una capa OSI superior) o quitar una capa de protocolo (función de Receptor: toma un paquete proveniente de una capa OSI inferior).

- Puertos de Entrada.

  In0 Puerto de ingreso del flujo de paquetes a procesar.

- Puertos de Salida.

  Out0 Puerto de salida de los paquetes procesados.

  Out1 Puerto de salida de la señal de petición de un nuevo paquete (indicando que el servidor está ocioso).

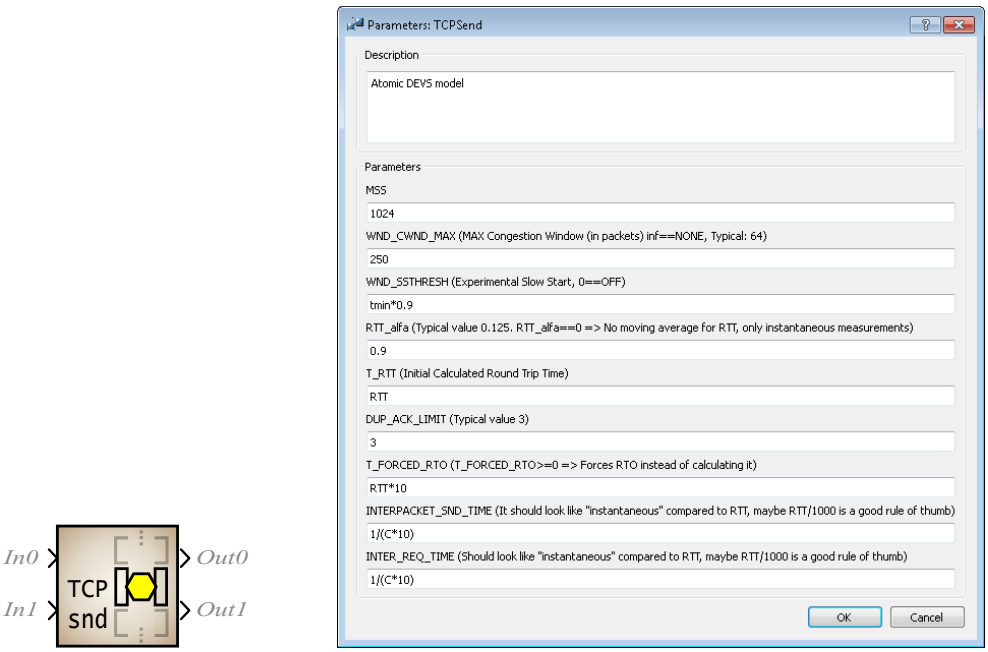
- Interfaz de Usuario.

MinTransitDelay Determina una cantidad mínima de demora constante (overhead) independiente de la longitud de cada paquete. (en milisegundos).

Bandwidth Ancho de banda (o capacidad) del servidor que determinará el tiempo de procesamiento proporcional a la longitud de cada paquete. (en bits por segundo).

LayerFunction Tipo de función en relación al tratamiento de la estructura de protocolos de cada paquete. TX=Transmisor en el stack OSI. RX=Receptor en el stack OSI.

B.1.7. TCP Transmisor



Ícono Interfaz de usuario

Figura B.7: TCPSend.

- **Descripción Funcional.**

  Aplica el mecanismo de control de congestión por ventana deslizante del protocolo TCP. (Siguiendo [SW94] y [SW95])

- Puertos de Entrada.

  In0 Puerto de ingreso del flujo de paquetes de DATOS a transmitir (desde capa superior de protocolo).

  In1 Puerto de ingreso de los paquetes tipo ACKNOWLEDGE provenientes del bloque TCP Receptor (al otro extremo de la comunicación).

- Puertos de Salida.

  Out0 Puerto de salida de los paquetes de datos (tasa regulada mor el mecanismo interno de sliding window).

  Out1 Puerto de salida de la señal de petición de un nuevo paquete a la capa de protocolo superior.

- Interfaz de Usuario.

  MSS TCP Maximum Segment Size.

  WND_CWND_MAX TCP Maximum Congestion Window.

  WND_SSTHRESH TCP Slow Start Threshold.

RTT_alfa Coeficiente para el cálculo de promedio móvil (moving average) para el cálculo del TCP RTT (Round Trip time).

T_RTT Valor inicial a asumir para RTT antes de obtener la primer medición efectiva.

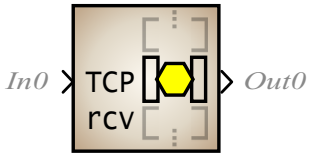
DUP_ACK_LIMIT Cantidad de ACK duplicados consecutivos que determinan indicación de congestión.

T_FORCED_RTO Valor para forzar el TCP RTO (Time Out) en lugar de calcularlo internamente como una función de RTT.

INTERPACKET_SND_TIME Valor heurístico práctico para determinar el tiempo de demora mínima entre la finalización del envío de un paquete y el próximo.

INTER_REQ_TIME Valor heurístico práctico para determinar el tiempo de demora mínima entre un pedido y el siguiente de nuevos paquetes a transmitir (a la capa superior).

B.1.8. TCP Receptor



Ícono

Figura B.8: TCPReceive.

- **Descripción Funcional.**

  Aplica el mecanismo de control de congestión por ventana deslizante del protocolo TCP, señalizando con paquetes ACK la recepción del flujo entrante. (Siguiendo [SW94] y [SW95])

- Puertos de Entrada.

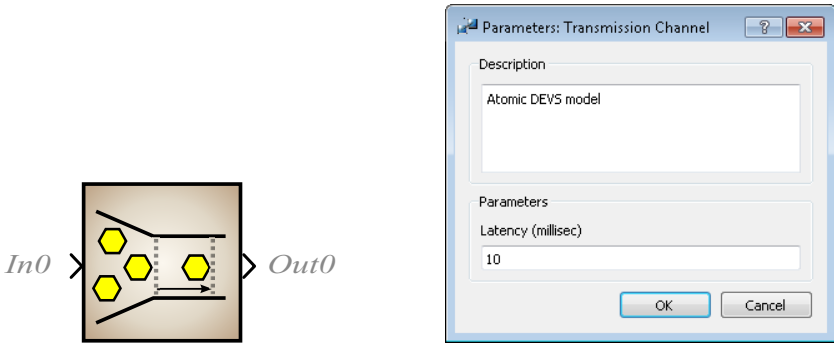
  In0 Puerto de ingreso del flujo de paquetes de DATOS proveniente desde el bloque TCP Transmisor (al otro extremo de la comunicación).

- Puertos de Salida.

  Out0 Puerto de salida de los paquetes de tipo ACKNOWLEDGE para ser consumidos por el bloque TCP Receptor (al otro extremo de la comunicación).

- Interfaz de Usuario.

No tiene.



Ícono Interfaz de usuario

Figura B.9: TXChannel.

B.1.9. Canal de Transmisión

- **Descripción Funcional.**

  Aplica el retardo propio de un canal físico de transmisión (capa física).

- Puertos de Entrada.

  In0 Puerto de ingreso del flujo de paquetes.

- Puertos de Salida.

  Out0 Puerto de salida de los paquetes de datos retrasados en el tiempo.

- Interfaz de Usuario.

Latency Latencia (demora) propia del canal físico.

B.1.10. Control de Admisión. Descarte Temprano y Aleatorio de Paquetes.

- **Descripción Funcional.**

  Aplica el mecanismo de descarte temprano aleatorio de paquetes (Random Early Detection) acorde a una función de probabilidad modulada por la longitud de la cola, propio de lo utilizado en las interaces de alta capacidad en los enrutadores.

- Puertos de Entrada.

  In0 Puerto de ingreso del flujo de paquetes.

  In1 Puerto de ingreso de la información de la longitud de la cola que se está controlando.

- Puertos de Salida.

  Out0 Puerto de salida de los paquetes que no fueron descartados.

  Out1 Puerto de información indicando la cantidad de paquetes descartados hasta el momento.

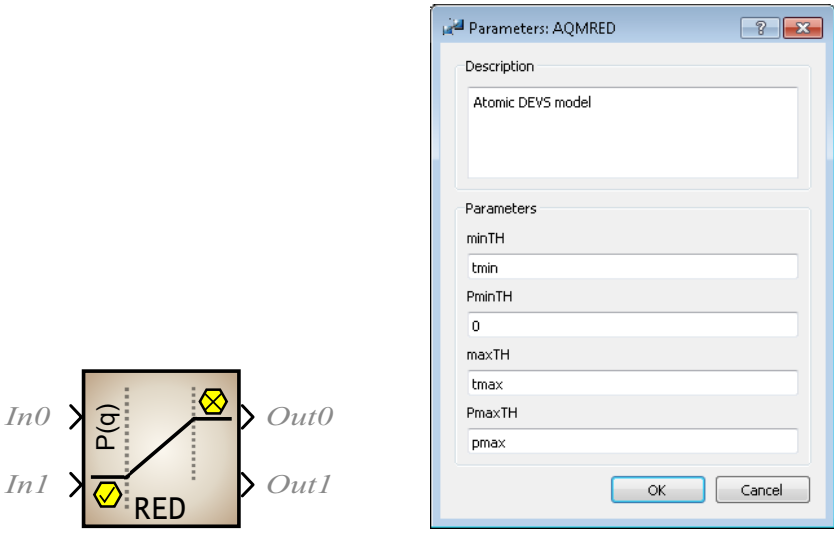


Figura B.10: AQMRED.

- Interfaz de Usuario.

minTH Umbral mínimo de longitud de cola para comenzar a descartar paquetes aleatoriamente. (en paquetes).

PminTH Probabilidad a la que se comienza a descartar cuando se alcanza el umbral mínimo. (entre 0 y 1).

maxTH Umbral máximo de longitud de cola a partir del cual la probabilidad de descarte satura en su valor máximo. (en paquetes).

PmaxTH Probabilidad máxima (de saturación) alcanzada al llegar la cola a maxTH. ($0 <= PminTH < PmaxTH <= 1$).

B.1.11. Control de Admisión. Token Bucket.

- **Descripción Funcional.**

  Aplica el mecanismo de control de ingreso de paquetes acorde al sistema de tickets (o tokens) Token Bucket. Cuando un paquete llega al bloque, será enviado por el puerto de salida sólo si existen tokens sin utilizar. Caso contrario se encola, y si se alcanza un límite de capacidad, se descarta. La generación de nuevos tokens se realiza internamente acorde a una tasa modulada externamente, funcionando como un mecanismo de regulación (limitación) de tasa de paquetes.

- Puertos de Entrada.

  In0 Puerto de ingreso del flujo de paquetes.

  In1 Puerto de ingreso para modular la tasa de generación de tokens.

- Puertos de Salida.

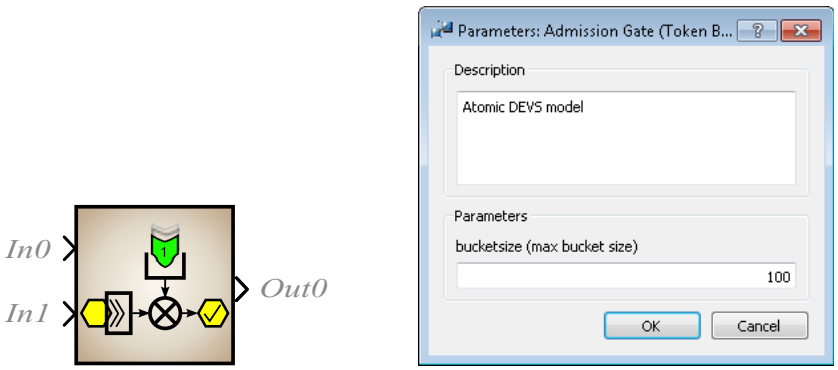


Figura B.11: TokenBucket.

Out0 Puerto de salida de los paquetes que no fueron descartados.

- Interfaz de Usuario.

bucketsize Tamaño máximo permitido para la cola interna de paquetes.

B.1.12. Combinador Híbrido de Flujos.

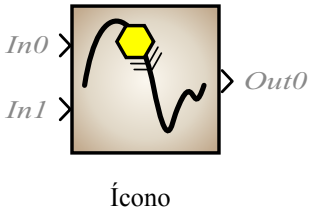


Figura B.12: HybridFlow.

- **Descripción Funcional.**

  Toma un flujo discreto de paquetes y una información continua de una longitud de cola. Con ello genera un nuevo flujo híbrido de salida, en el cual la longitud de cada paquete tomada del flujo discreto es incrementada para que ocupe una longitud tal que imite la longitud de cola indicada por la información continua.

  El bloque es detalladamente descrito en la Sección 7.4.2.

- Puertos de Entrada.

  In0 Puerto de ingreso del flujo discreto de paquetes.

  In1 Puerto de ingreso informando la longitud continua de una cola.

- Puertos de Salida.

Out0 Puerto de salida de los paquetes con formato híbrido.

- Interfaz de Usuario.

No tiene.