Charla de Borrachos

Modelos y simuladores en tiempo real para control de performance en redes de datos

Dr. Rodrigo Castro

Viernes 16 de Diciembre de 2011

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

UNIVERIDAD NACIONAL DE ROSARIO

Carleton UNIVERSITY

CIFASIS
CONICET
UNR | UPCAM

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Organización

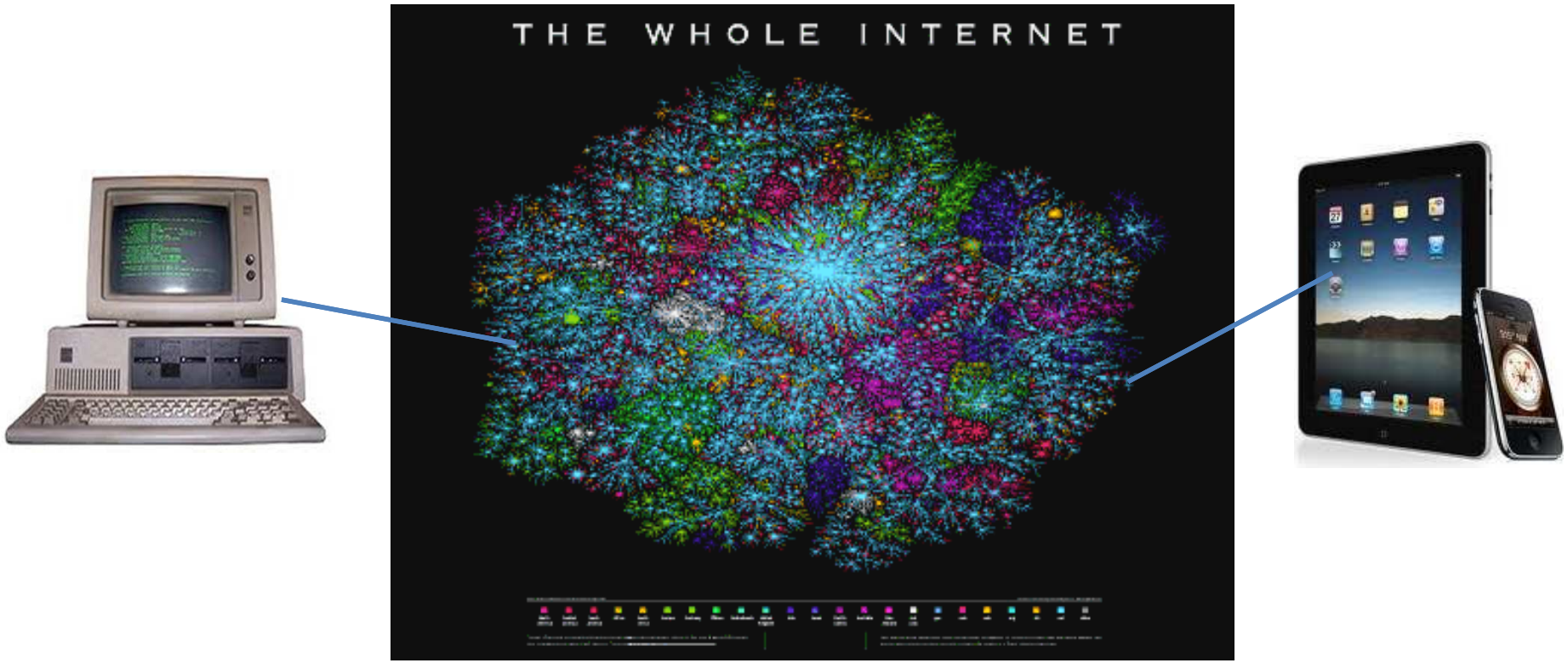
- Introducción
  - Escenario Motivador
    - Control jerárquicos de redes de datos
  - Presentación de problemas
- Metodología unificada basada en el Formalismo DEVS
- Parte 1) Aproximación mediante modelos fluidos
  - Control de Congestión de TCP
- Parte 2) Metodología para aplicar Teoría de Control
  - Control de Admisión de Paquetes
- Bonus

  Parte 3) Implementación directa de Controladores en Tiempo Real
  - Control Supervisorio basado en Medición de Tasa
- Conclusiones

- 2 temas en paralelo
  - La aplicación
    - Diseño de controladores
      - Proveer calidad de servicio en internet (sistema de software/físico)
  - La metodología
    - Métodos y Herramientas de Modelado y Simulación
      - Multiparadigma, Multiformalismo, Multiplataforma.
- Mensajes clave:
  - Metodología en capas
  - Coexistencia de controladores heterogéneos
  - Simulación embebida en tiempo real

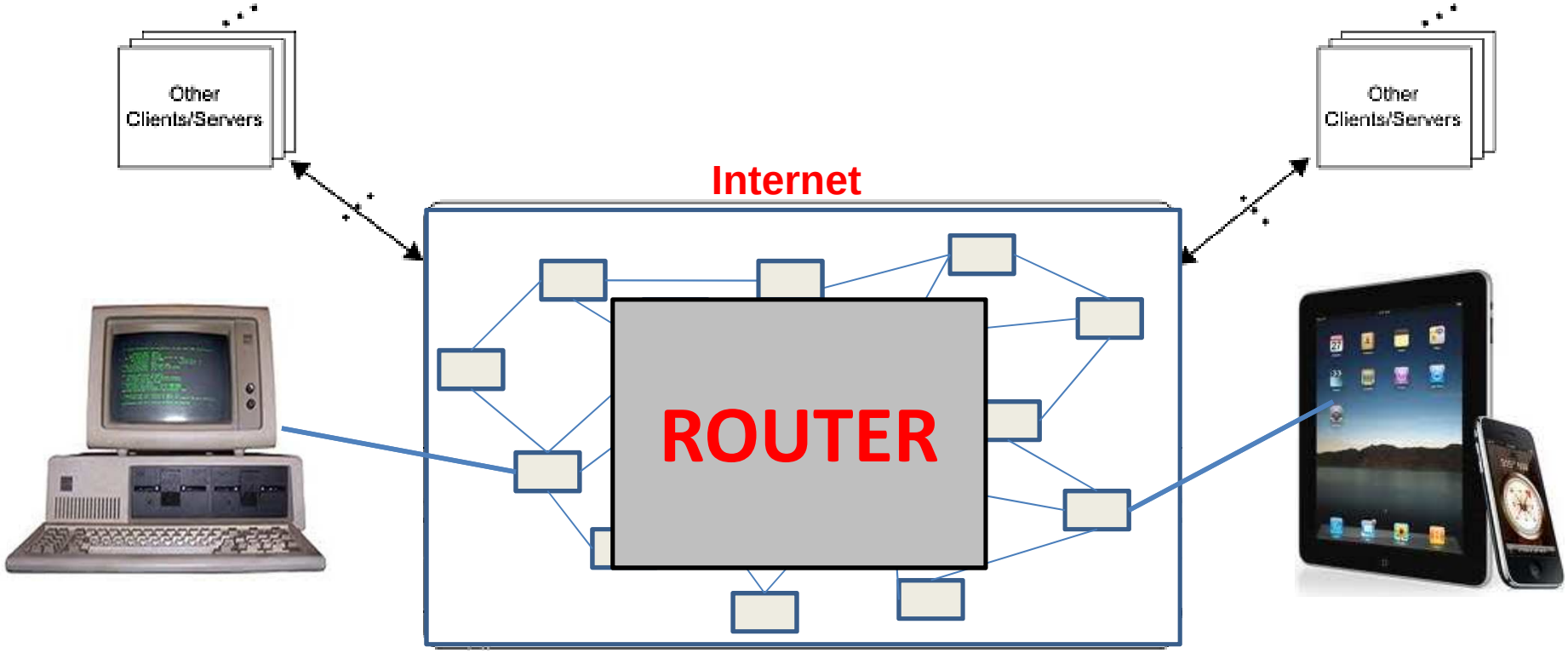
Escenario Motivador

- Internet
  - Modelo romántico



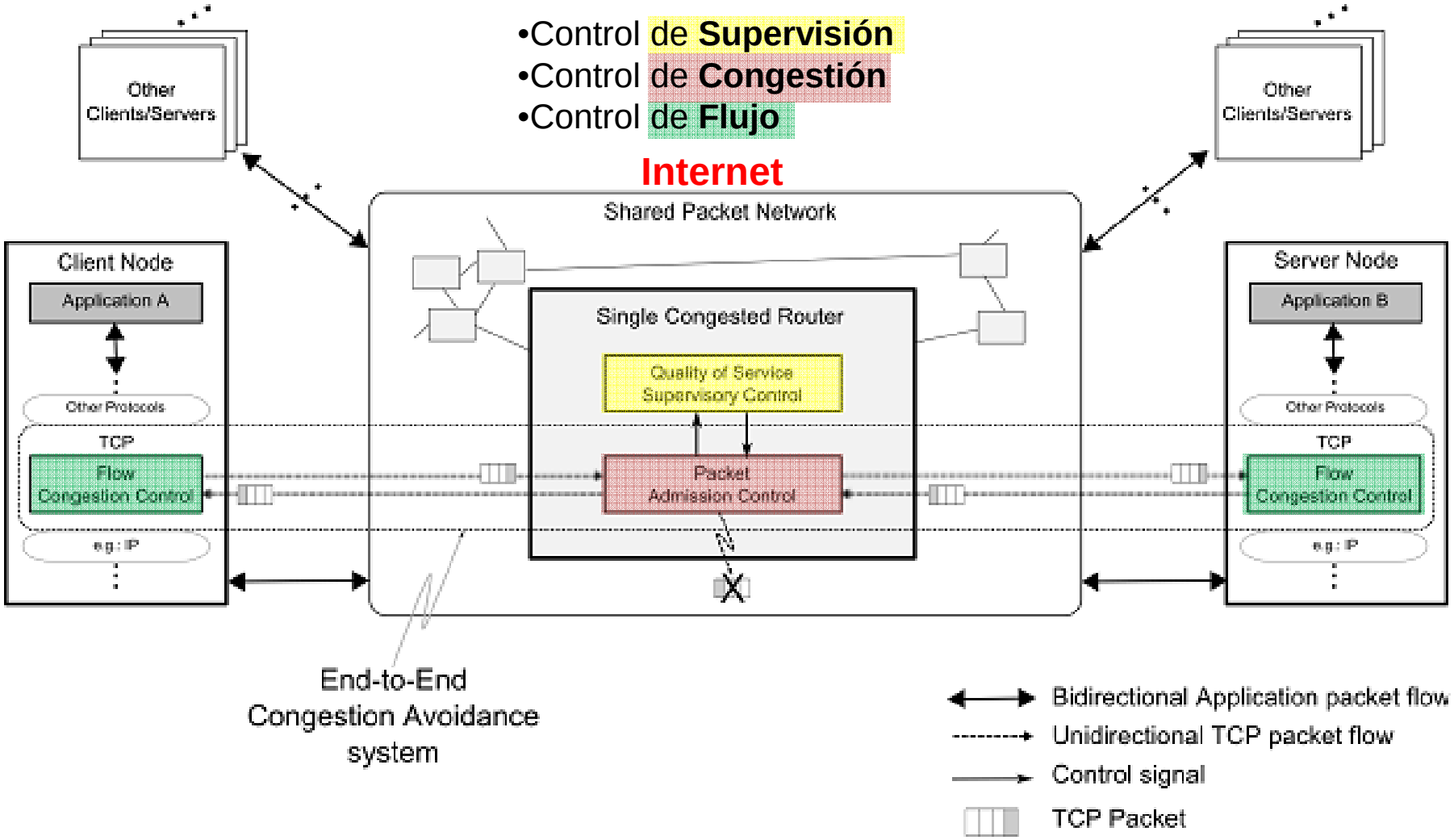
Escenario Motivador

- Internet
  - Un poco más de detalle…

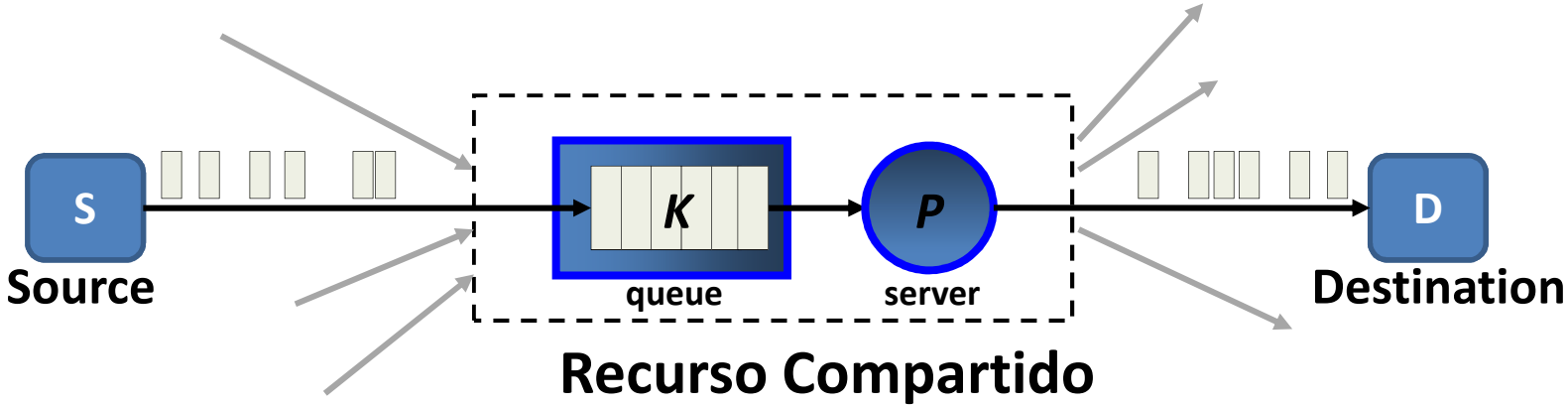


Escenario Motivador

- Internet
  - Suficiente detalle: 3 tipos de Control simultáneos



- Optimización de recursos en Redes de Datos
  - **Múltiples usuarios** compiten por **recursos finitos**



- Asignación eficiente de recursos: Diseño de controladores
  - A distintos niveles de granularidad
  - Promoviendo la aplicación de Teoría de Control
  - Regular los niveles de Calidad de Servicio que la red provee a los usuarios

- Características de la Red: Pueden ser cualquiera
  - Sistema Complejo
  - Recurrimos a **Metodologías de Modelado y Simulación**
  - Análisis, Diseño, Verificación, Validación e Implementación

- La comunidad de **Modelado y Simulación de redes** utiliza principalmente **modelos a Eventos Discretos**
  - Proveen **máximo nivel de detalle**, **pero no son suficientes**

Propuesta de solución

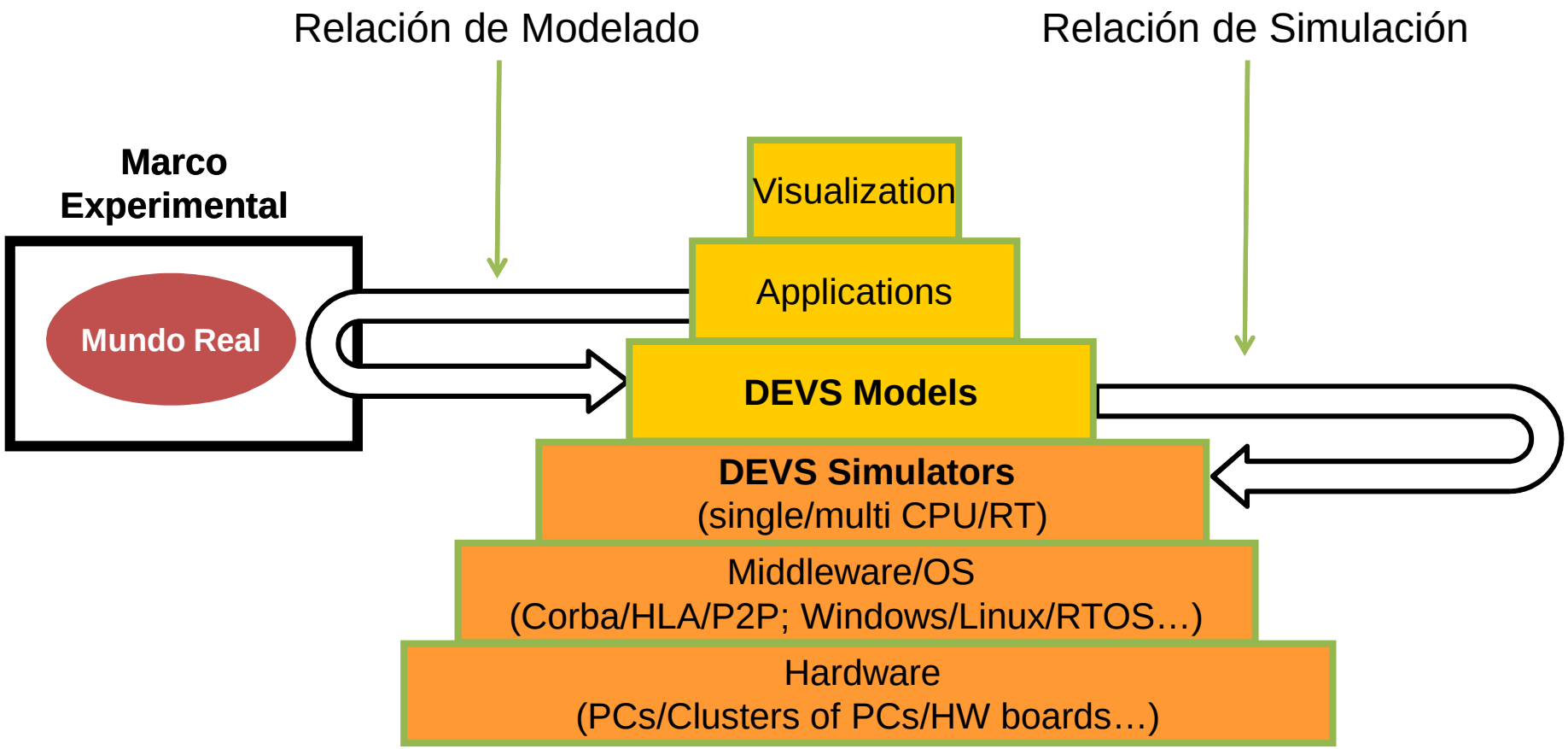
- Práctica Usual:
  - Formalismos y herramientas diferentes para cada necesidad
    - Ineficiente.
    - Propenso a cometer errores formales y prácticos.

- Propuesta de Solución
  - **Metodología unificada** basada en el **formalismo DEVS** para Modelado y Simulación de **Sistemas Híbridos**

- Un mismo formalismo y herramienta
    - Desde el **análisis** hasta la **implementación final** en hardware
    - **DEVS** permite el modelado simultáneo de diversos paradigmas
      - Modelos Continuos, a Eventos Discretos y a Tiempo Discreto

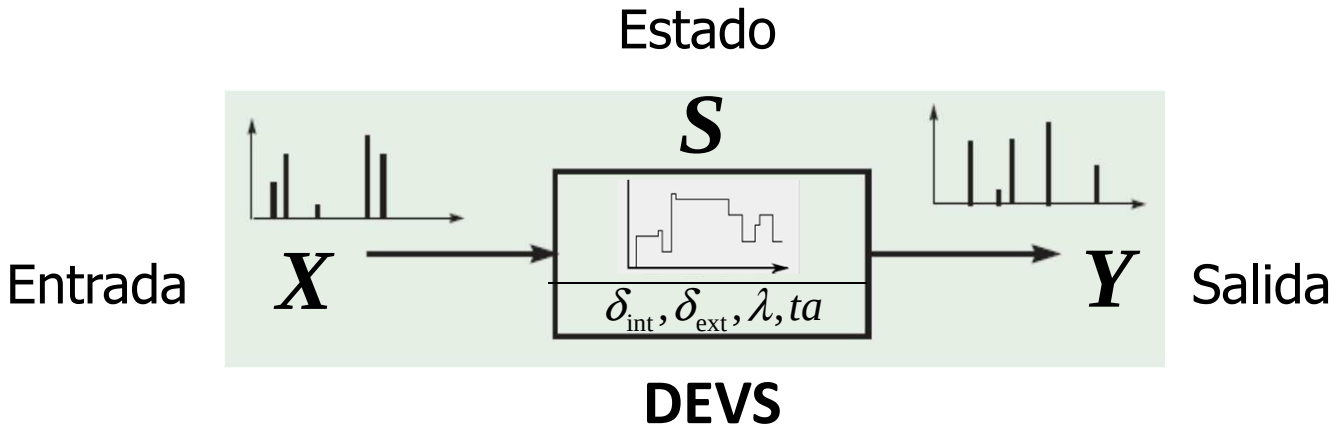
- **Metodología**
  - **Formalismo DEVS**
    para Modelado de Sistemas de Eventos Discretos Generalizados

- **M&S por Capas**


Relación de Modelado
Relación de Simulación
Marco Experimental
Mundo Real
Visualization
Applications
DEVS Models
DEVS Simulators
(single/multi CPU/RT)
Middleware/OS
(Corba/HLA/P2P; Windows/Linux/RTOS…)
Hardware
(PCs/Clusters of PCs/HW boards…)

- **DEVS** (Discrete EVent Systems specification, Bernard Zeigler, ’76)
- Basado en la Teoría General de Sistemas
- **DEVS** permite:
  - **Representar exactamente** cualquier **sistema discreto**
  - **Aproximar sistemas continuos** con tanta precisión como se desee

- Modelo Atómico



- Un modelo DEVS queda definido por la siguiente estructura:

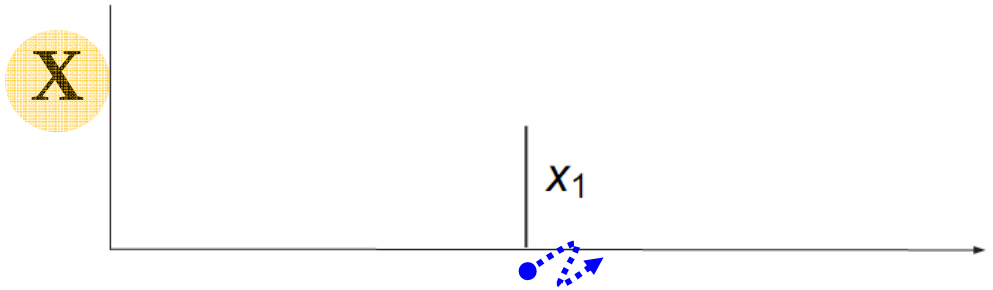
$$M_D = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta)$$

Conjuntos — Funciones Dinámicas

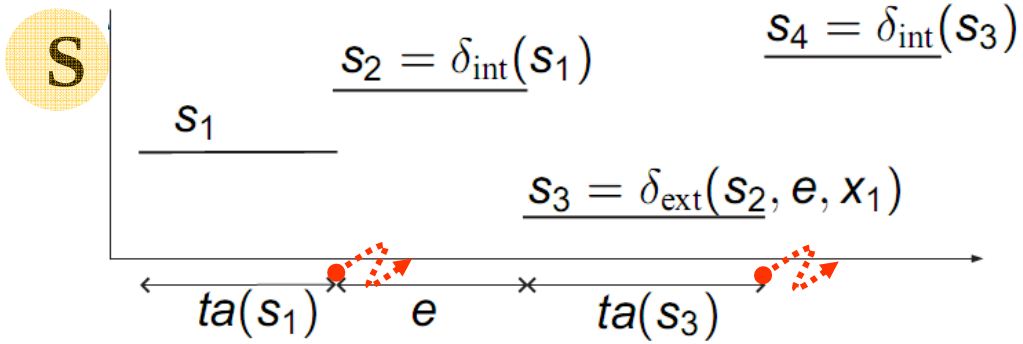
- Ejemplo

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

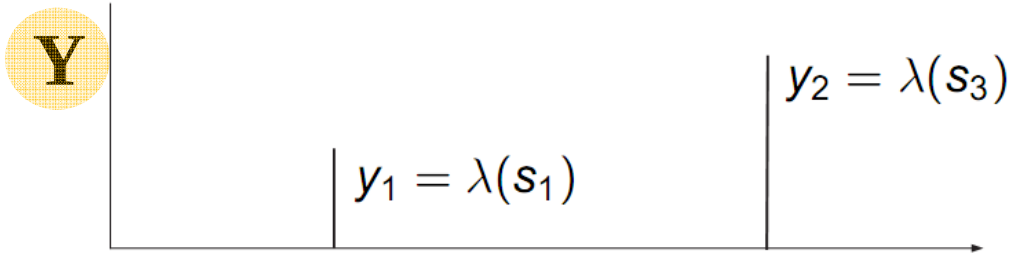
Trayectoria de Entradas



Trayectoria de Estados Internos

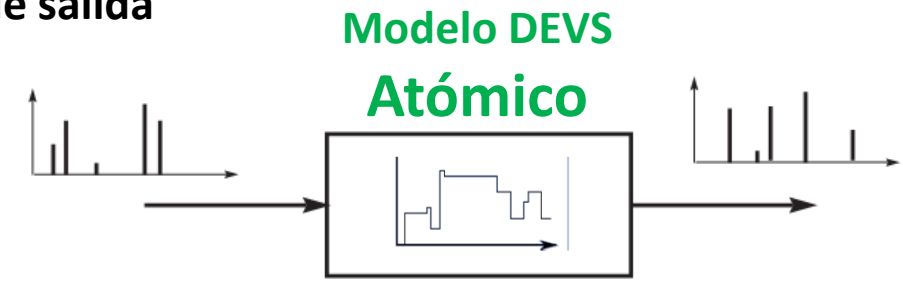


Trayectoria de Salidas



Propiedades

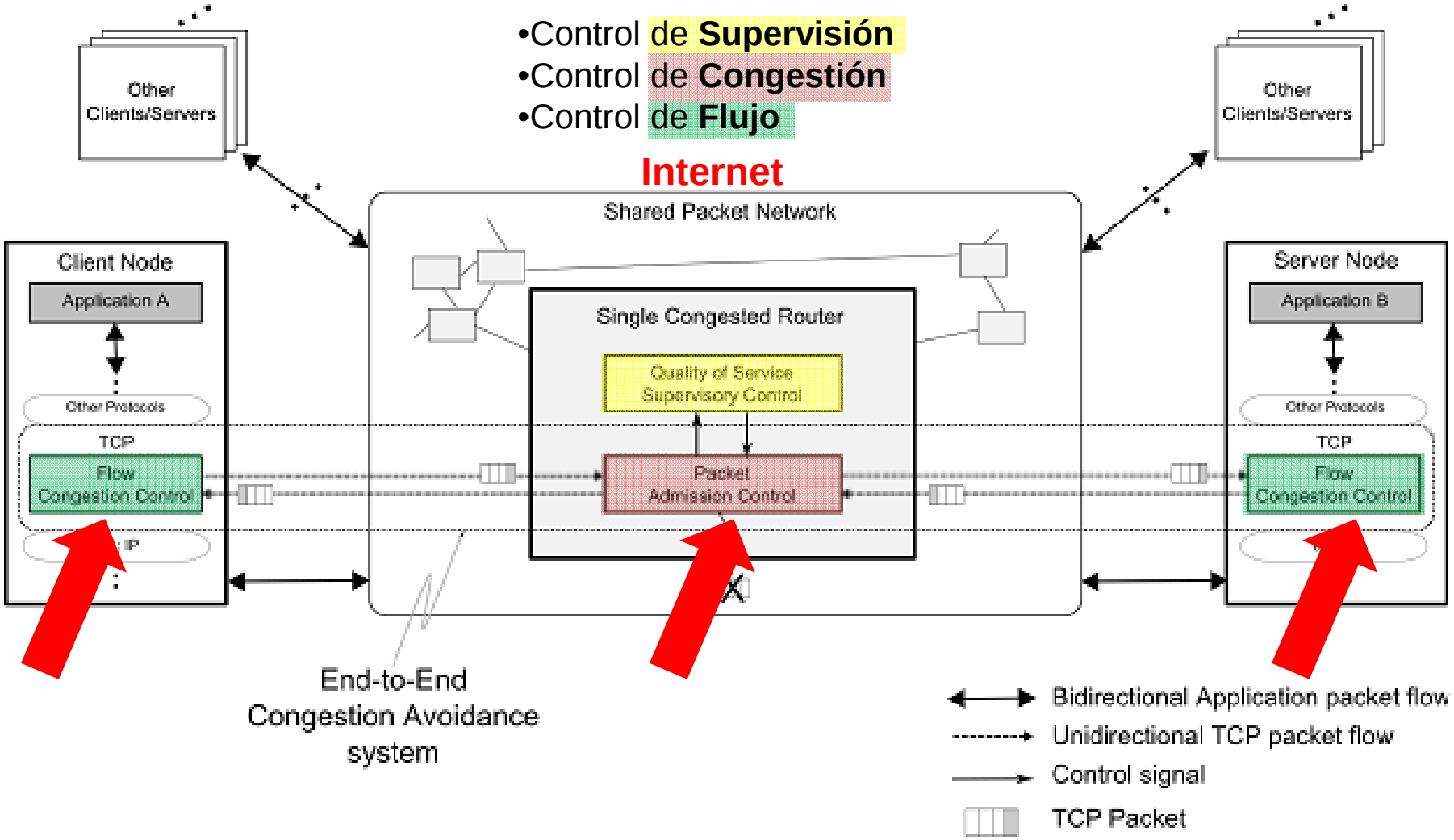
- Un modelo DEVS:
  - procesa una **secuencia de eventos de entrada**
  - de acuerdo a su cambio de **estado interno**
  - produce una **secuencia de eventos de salida**
- Utiliza una base de tiempo continuo



- Permite representar **cualquier sistema** que tenga un **número finito de cambios** en un **intervalo finito de tiempo**
  (Modelos Legítimos)

- Aproximación mediante modelos fluidos
  - **Control de Congestión en TCP**

- Aproximación mediante modelos fluidos
  - **Control de Flujo y Congestión en TCP**



- **Control de Flujo y Congestión en TCP**

Mecanismo de Ventana Deslizante

+

Descarte Aleatorio de paquetes

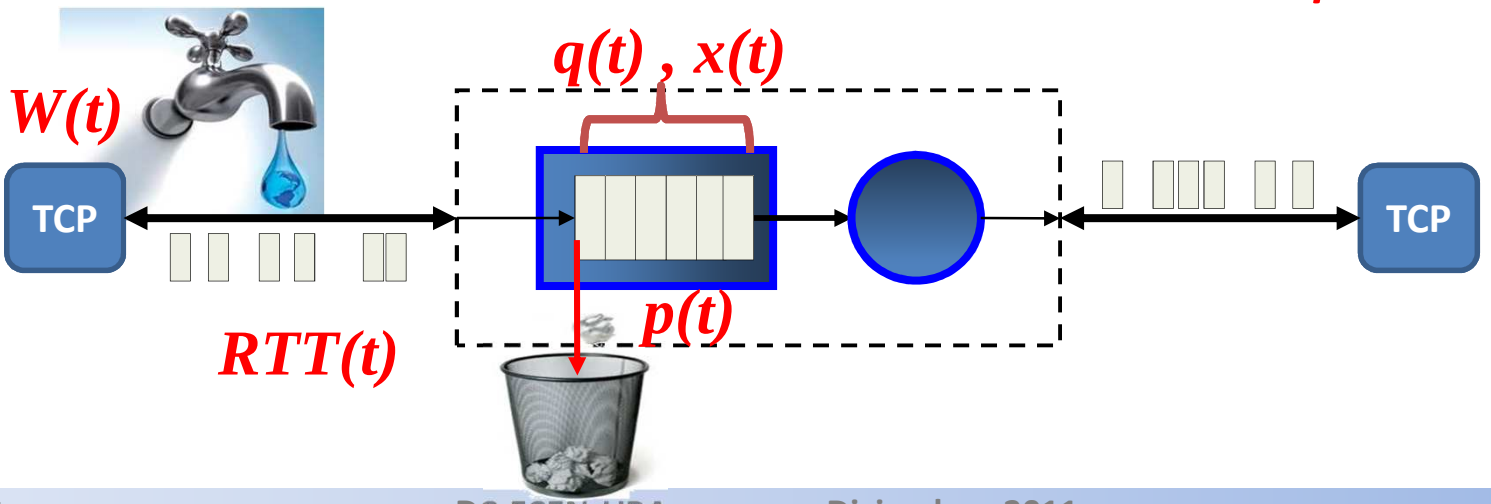
Ecuaciones diferenciales

- Un modelo fluido del **Control de Congestión en TCP**
- Sistema de Ecuaciones Diferenciales con Retardo:

$$\dot{W}(t) = \overbrace{\frac{1}{RTT(q(t))}}^{AI} - \overbrace{\frac{W(t)}{2}}^{MD} \times \frac{W(t-\tau)}{RTT(q(t-\tau))} p(x(t-\tau))$$

$$\dot{q}(t) = -1_{q(t)} C + N \times \frac{W(t)}{RTT(q(t))}$$

retardo dependiente del **estado *q***



Particularidades

- De Modelado y Simulación
  - Las redes imponen **demoras variables** en los modelos fluidos, produciendo **ecuaciones diferenciales con retardo**

    - Los métodos numéricos que permiten a DEVS aproximar ecuaciones diferenciales:

      QSS

      ahora también

      pueden manipular señales continuas retardadas

Herramienta previa: QSS

- Métodos Numéricos **Quantized State Systems (QSS)**
- Idea Básica:
  - **Cuantificar las variables de estado en vez de particionar el tiempo en pasos discretos**
  - Mantener el eje de tiempo continuo

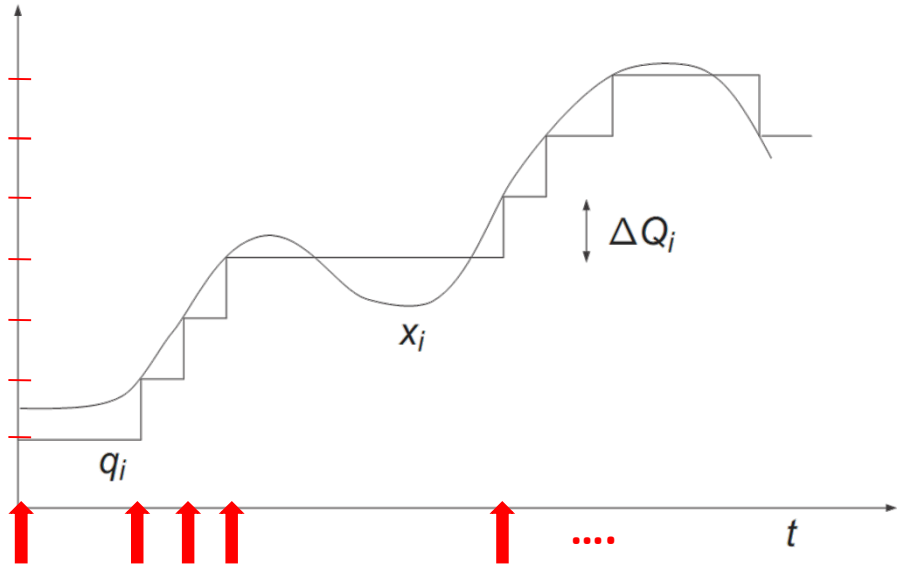
Herramienta previa: QSS

- Consideremos un sistema de Ecuaciones Diferenciales Ordinarias (**ODE**) de orden **n**:

$$\dot{x}_1 = f_1(x_1(t),\cdots, x_n(t),t)$$
$$\vdots$$
$$\dot{x}_n = f_n(x_1(t),\cdots, x_n(t),t)$$

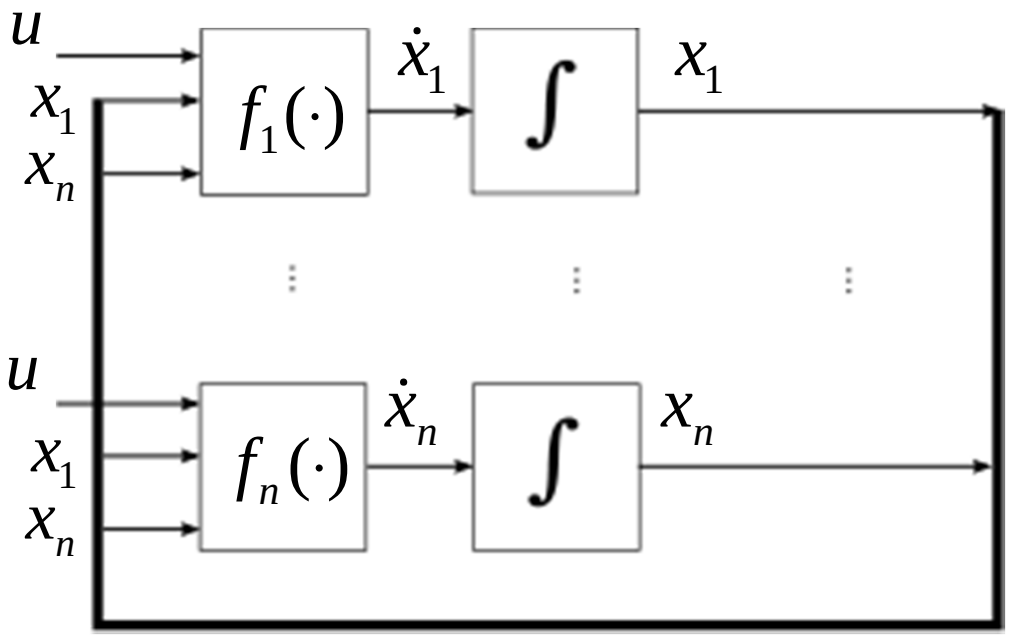
- **Al cuantificar $x_1,\ldots,x_n$** (variables de estado) se obtiene:

$$\dot{x}_1 = f_1(q_1(t),\cdots, q_n(t),t)$$
$$\vdots$$
$$\dot{x}_n = f_n(q_1(t),\cdots, q_n(t),t)$$



Herramienta previa: QSS

- Representación de un sistema ODE



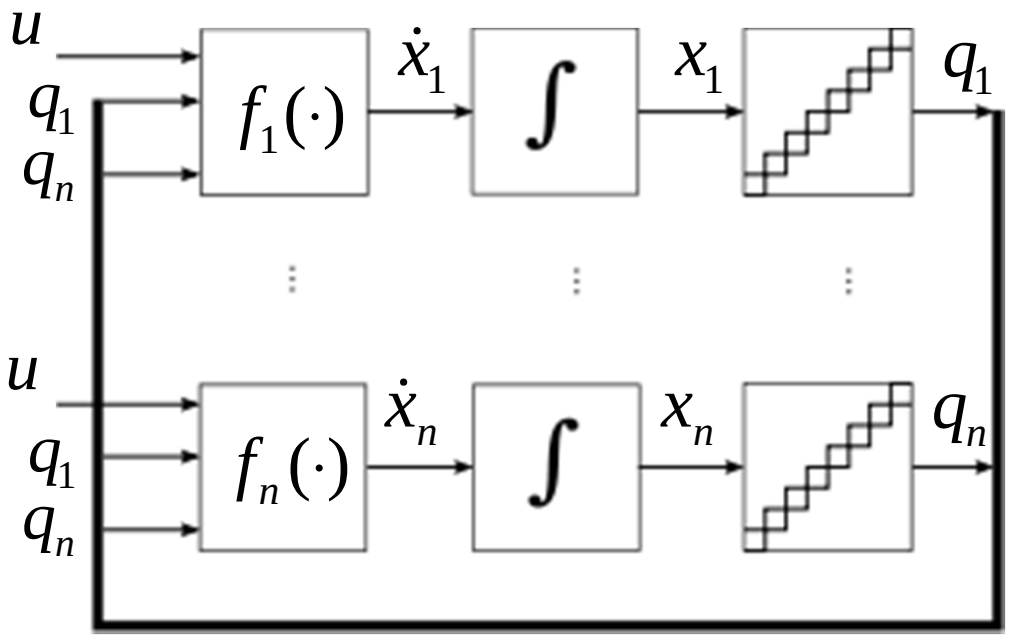
$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$$

Sistema Original

(exacto)

Herramienta previa: QSS

- Representación de un sistema QSS



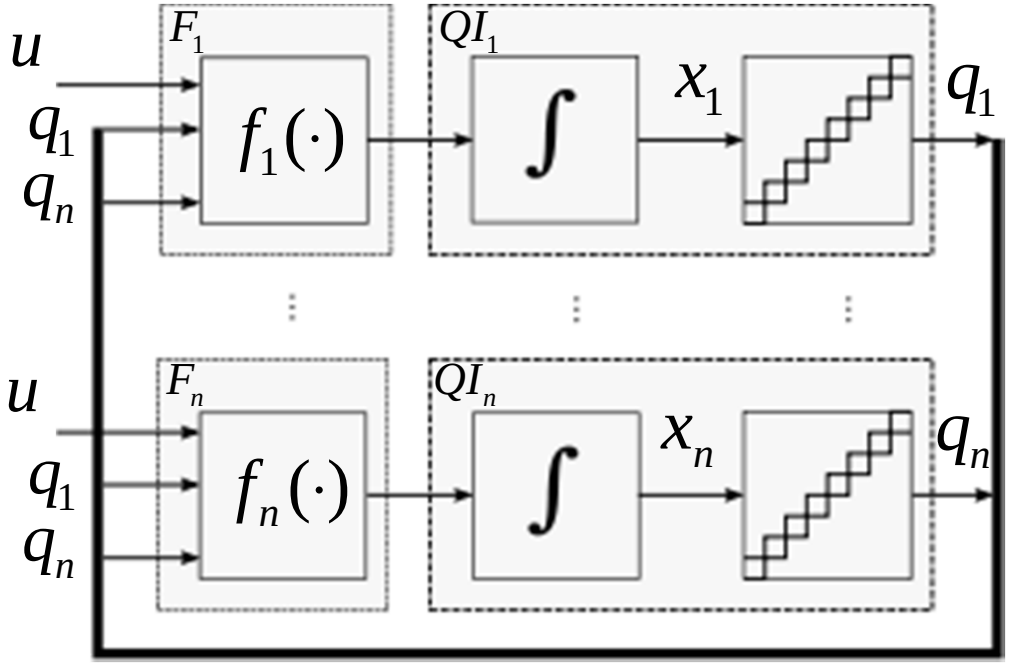
$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$$

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

Sistema Cuantificado
(aproximación)

Herramienta previa: QSS

- Representación mediante un sistema DEVS

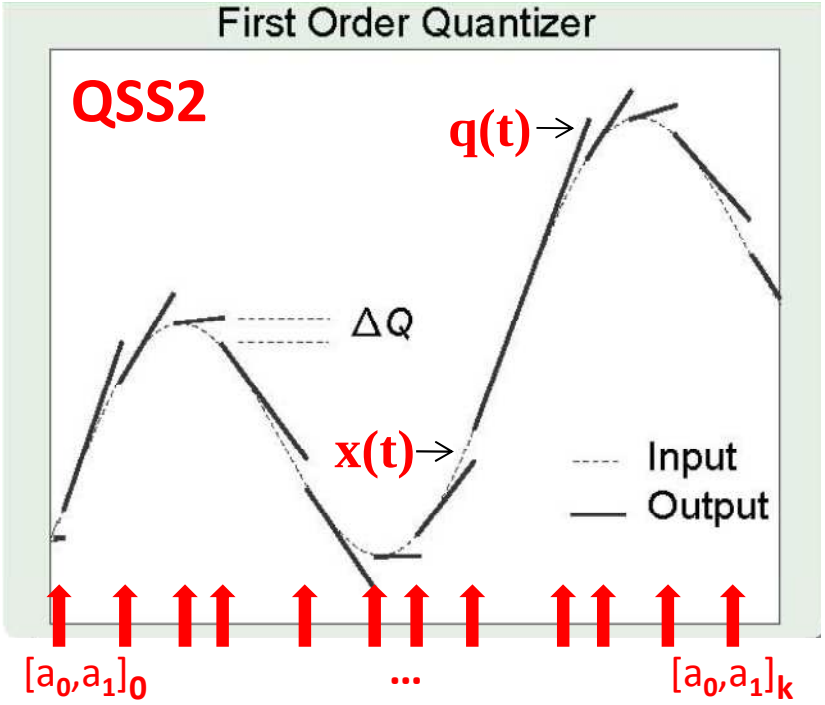


$$\dot{\mathbf{x}}(t) = f(\mathbf{q}(t), \mathbf{u}(t))$$

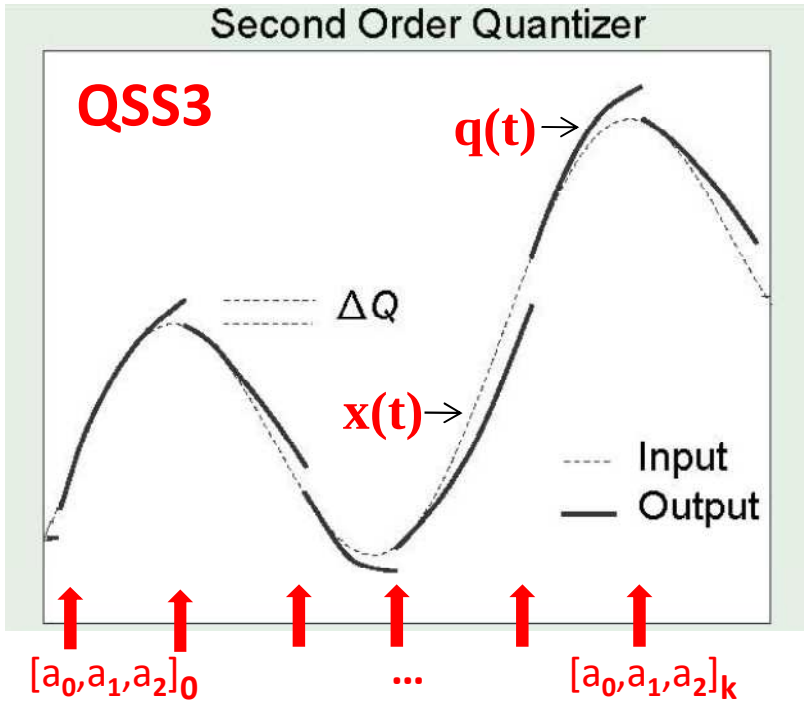
Equivalente DEVS

Herramienta previa: QSS

- Desventaja de QSS
  - Ineficiente (Método de primer orden)
  - El **número de pasos** crece linealmente con la precisión
- Solución: QSS de órdenes superiores



- Método de Segundo Orden
- **Número de pasos** crece con la **raíz cuadrada** de la precisión



- Método de Tercer Orden
- **Número de pasos** crece con la **raíz cúbica** de la precisión

Ventajas QSS

- Intrínsecamente asíncronos
  - Cambio desacoplado de variables de estado
- Salida densa sobre una base de tiempo continua
  - Detección y tratamiento eficientes de discontinuidades
- Preservan estabilidad práctica
- Puede calcularse una cota de error global de integración
- Especialmente aptos para: simulación en tiempo real, representación de sistemas híbridos

Delay QSS

- **Nuevo: Delay Quantized State Systems (DQSS)**

- Métodos numéricos para resolver **Delay Differential Equation (DDE)** basados en QSS
- QSS provee salida densa: puede aprovecharse para el cálculo (interpolación) de **señales retardadas**

Formulación

- Representación de un sistema DDE:

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

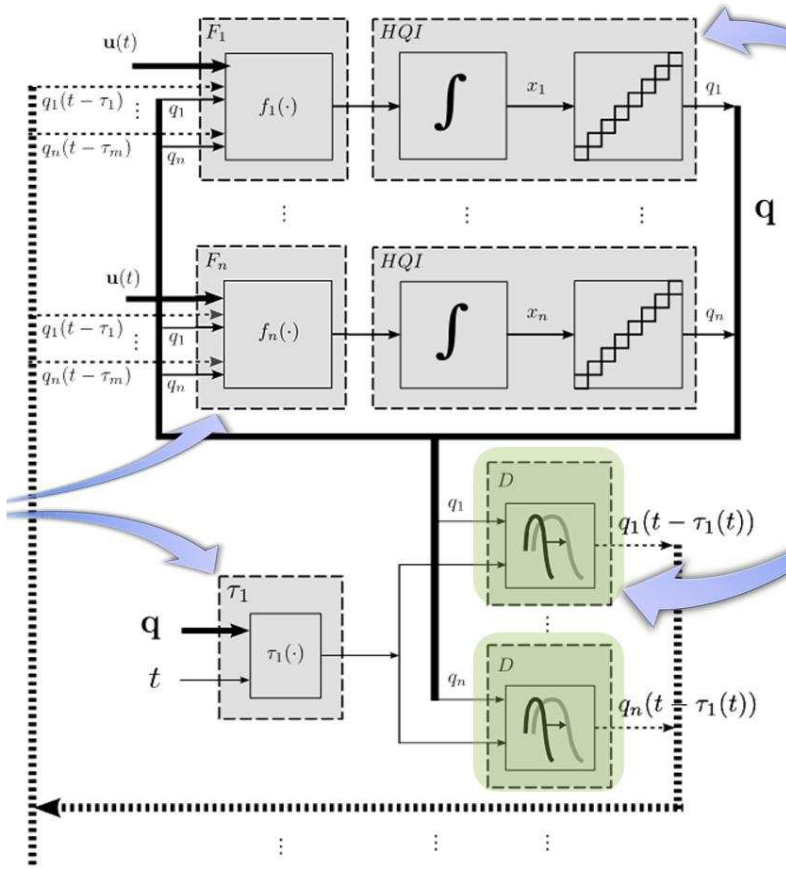
- Ahora seguimos la misma idea de cuantificación aplicada en QSS

Equivalente DEVS

- Representación mediante un sistema DEVS

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

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



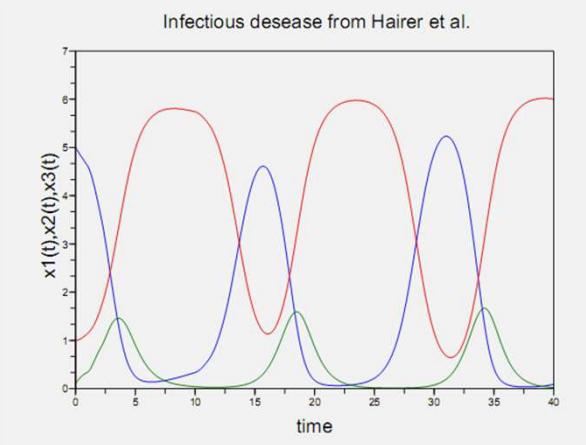
Integrador Cuantificado

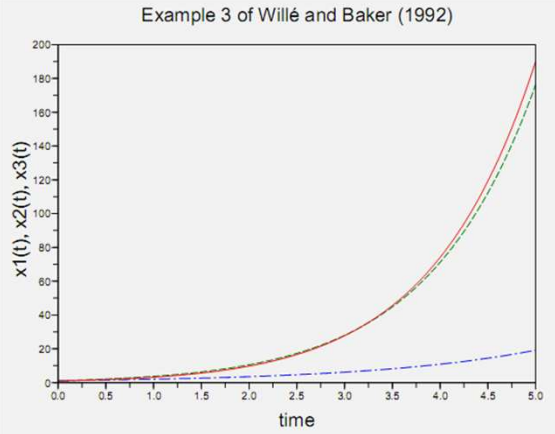
Nuevo Bloque: Delay Variable

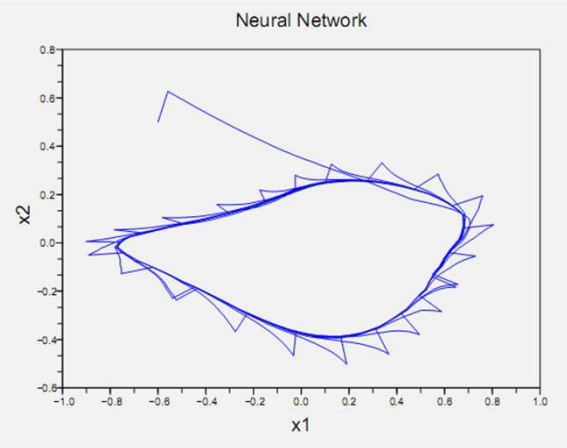
Función Estática

Propiedades DQSS

- Permite aproximar todos los tipos de DDE existentes
  - de delay constante, variable, dependiente del estado y neutral.
- Implementado hasta 3er. Orden de precisión (DQSS3)
- Hereda todas las propiedades de QSS
- Modelado intuitivo, sin necesidad de realizar análisis cualitativos previos del sistema.
- Tiempos de simulación hasta **7 veces más rápido** respecto de **dde23** (Runge Kutta/Matlab) para ejemplos de la literatura







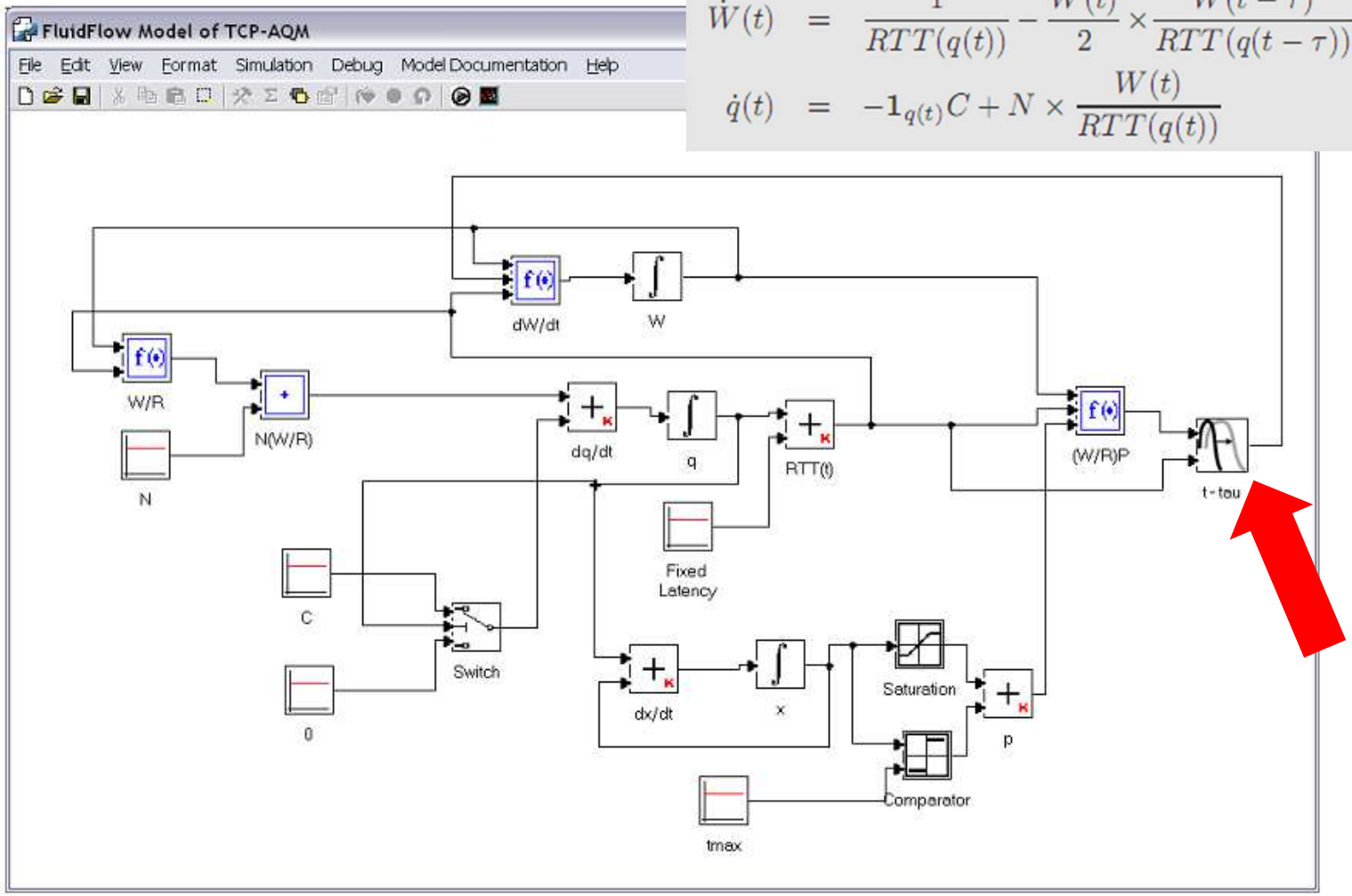
- ***DQSS trasciende el dominio de aplicación de redes de datos.***

Modelo continuo

- Estudio inicial mediante aproximación fluida

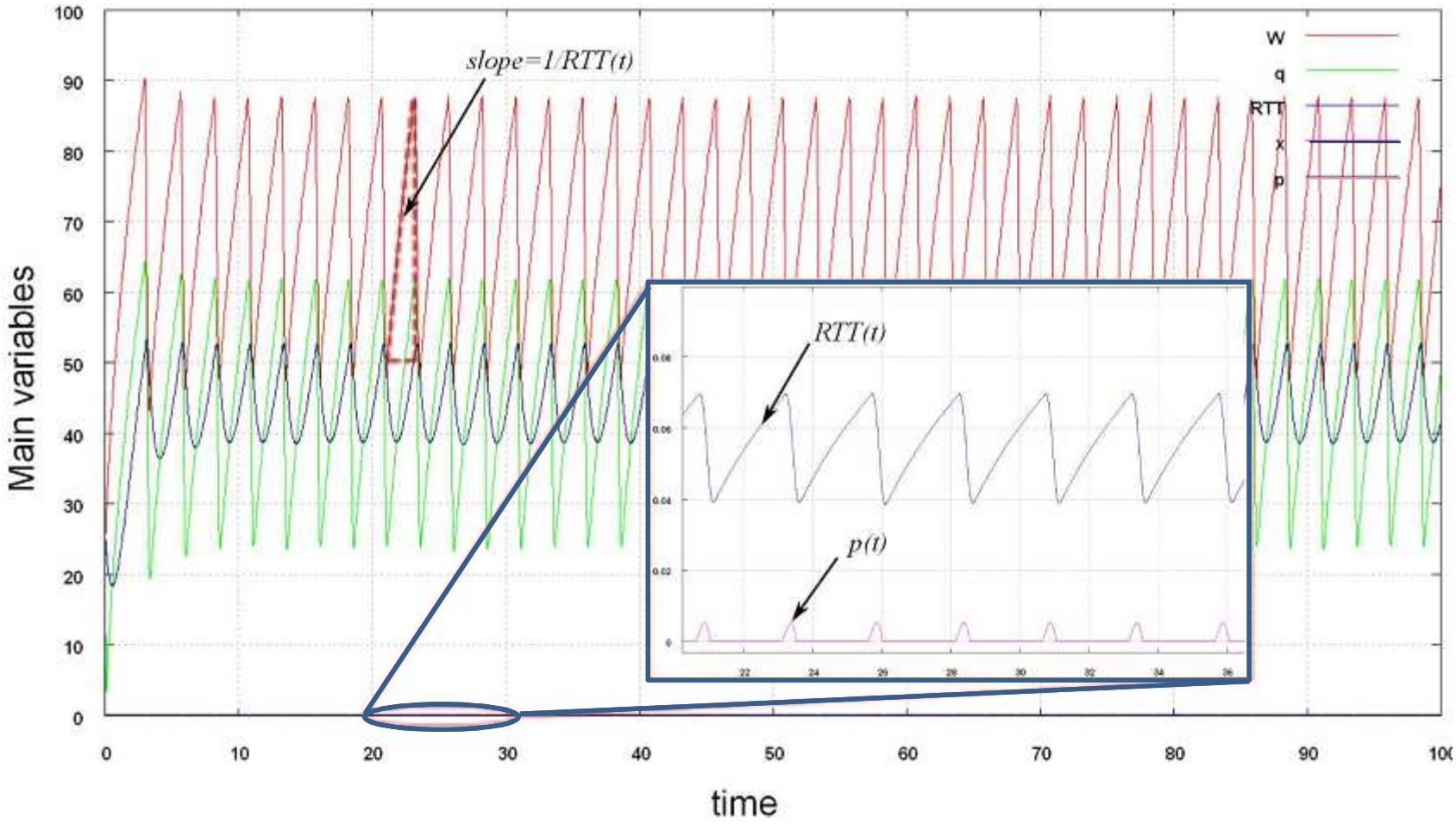
$$\dot{W}(t) = \overbrace{\frac{1}{RTT(q(t))}}^{AI} - \overbrace{\frac{W(t)}{2}}^{MD} \times \frac{W(t-\tau)}{RTT(q(t-\tau))} p(x(t-\tau))$$

$$\dot{q}(t) = -\mathbf{1}_{q(t)} C + N \times \frac{W(t)}{RTT(q(t))}$$



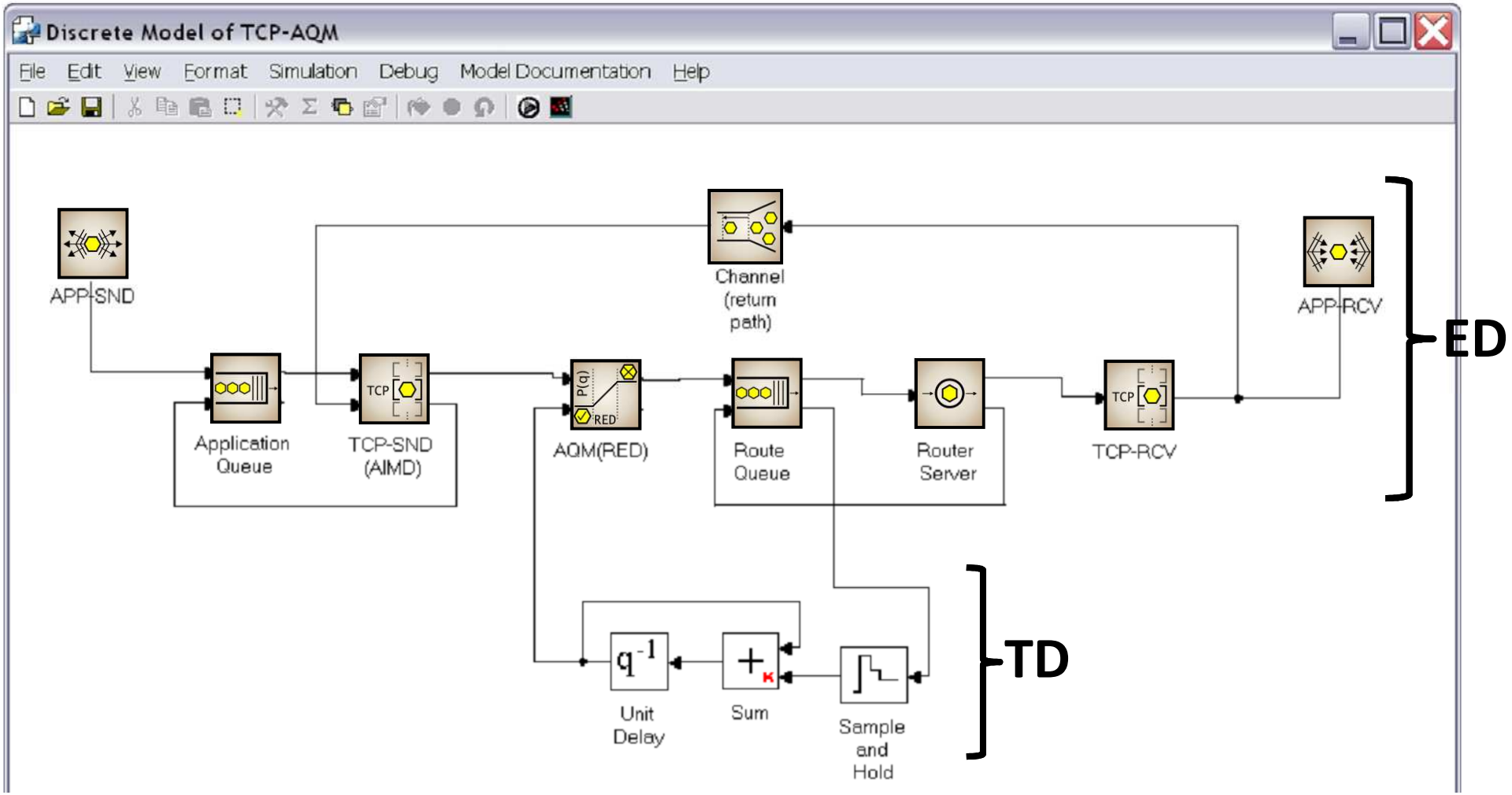
Resultados

- Simulación del modelo continuo



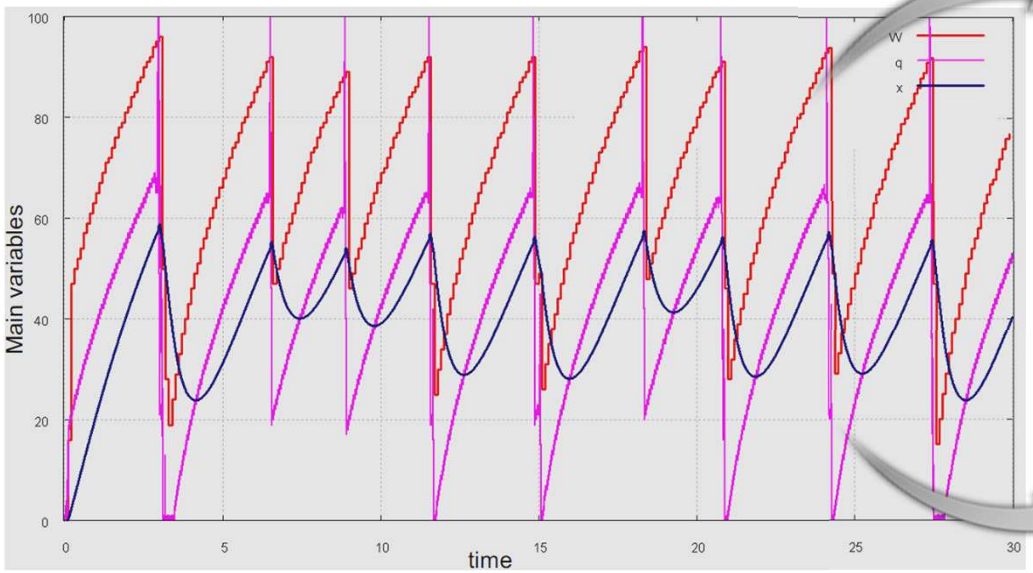
Modelo discreto

- Estudio detallado (primeros principios)

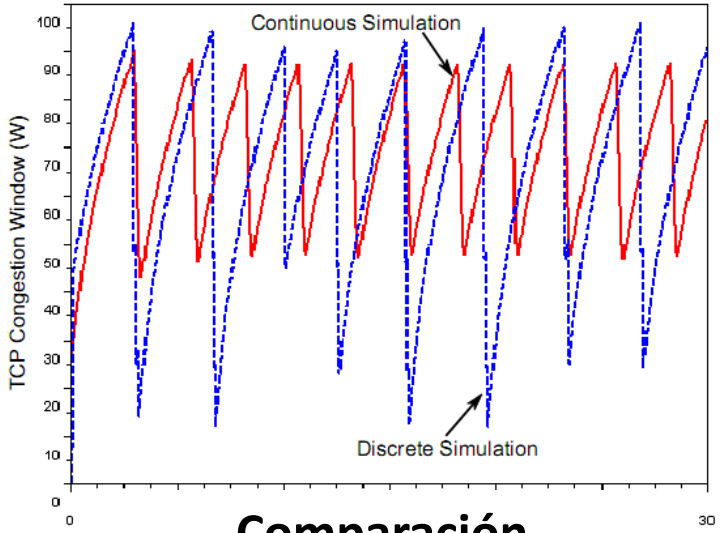


Resultados

- Verificación mediante simulación del modelo discreto

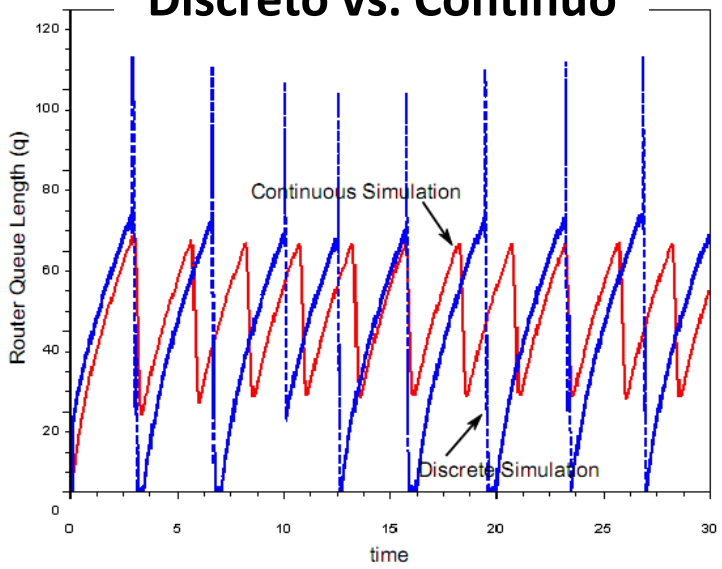


W



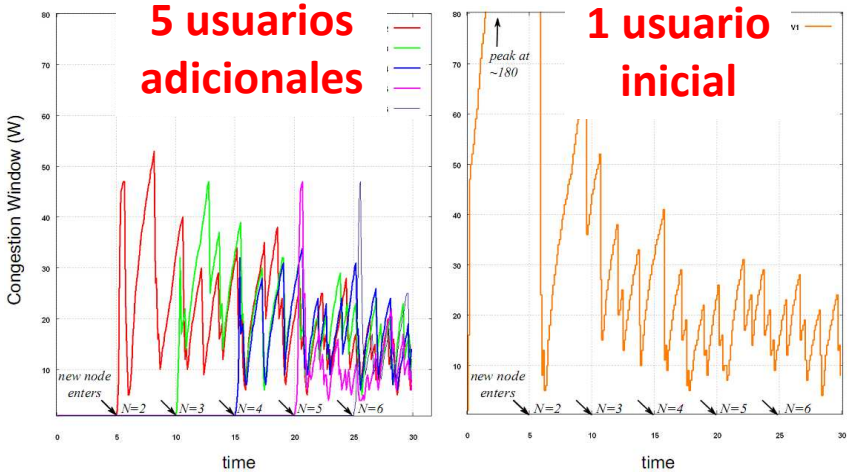
Comparación Discreto vs. Continuo

q

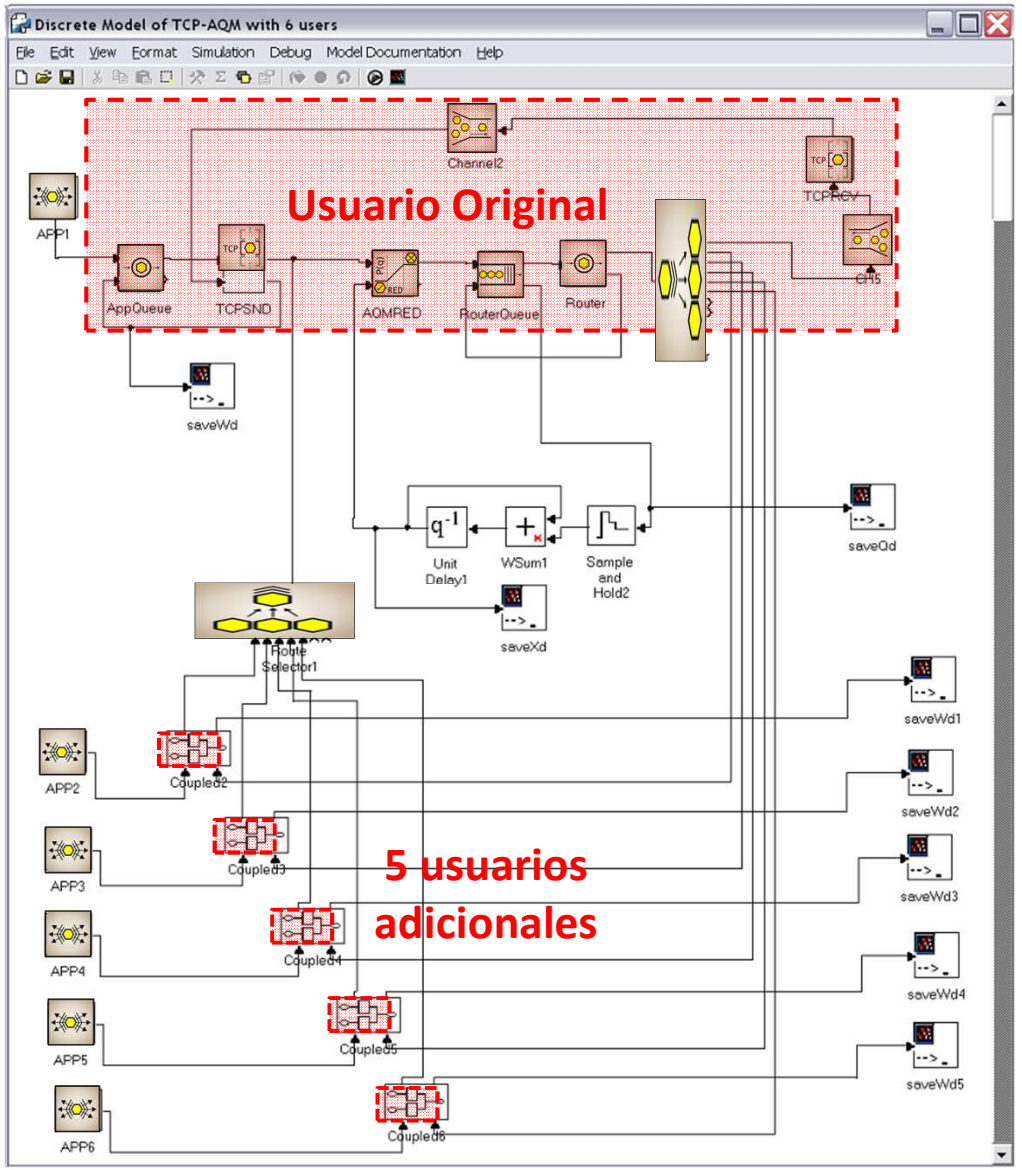


Múltiples usuarios

- Modelo **Discreto puro**
  - **Inconveniente** e **Ineficiente** para modelar explícitamente muchos flujos concurrentes



- **27 veces más lento** que el modelo fluido (para N=6)

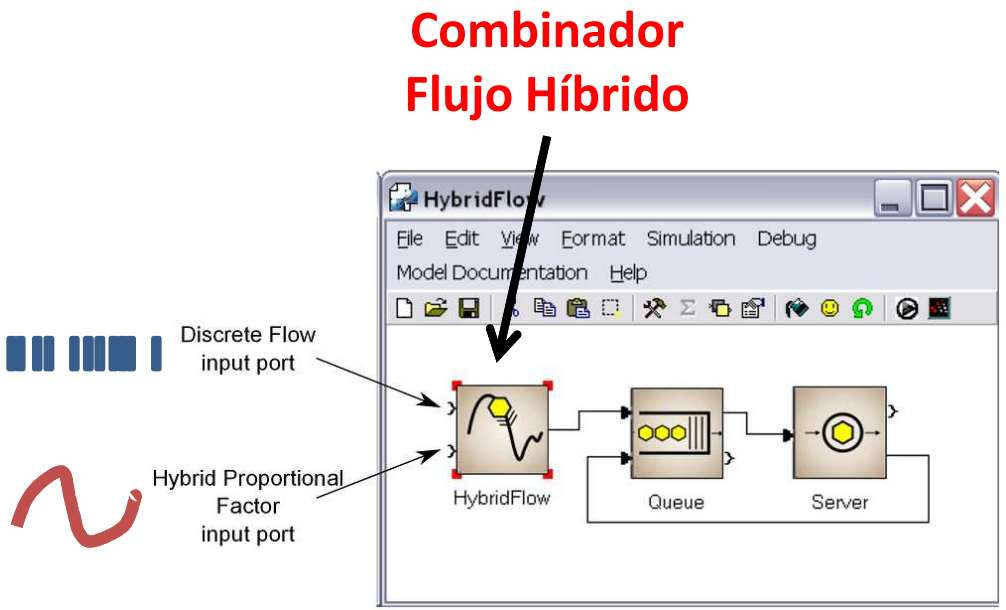


Flujo Híbrido

- **Solución:**
  Flujo Híbrido
  Discreto+Continuo
  1 Flujo discreto
  "de testigo"

- El nivel de llenado de la cola es indicado por la **parte continua** del sistema

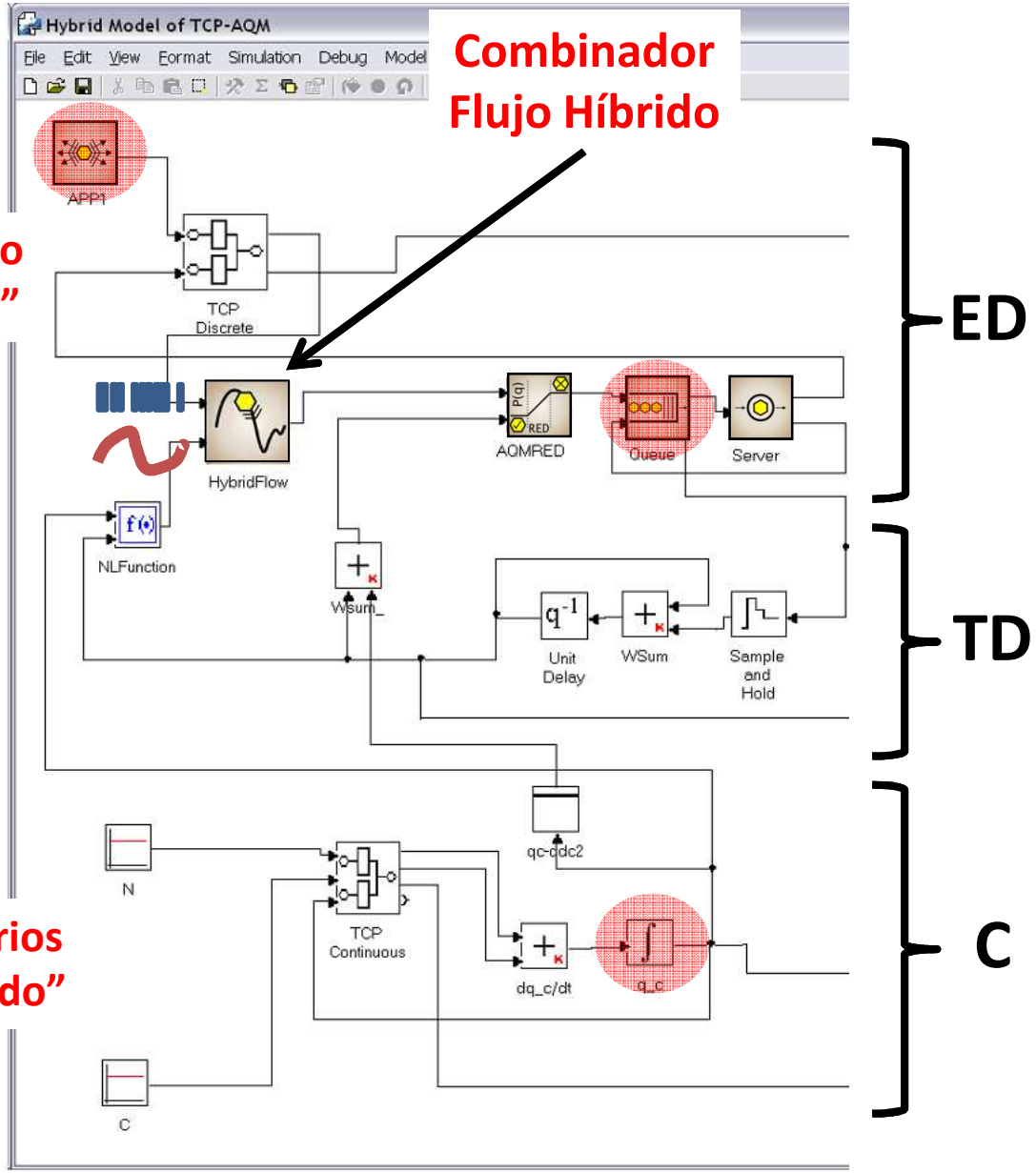
- La parte "híbrida" se almacena en la estructura de cada paquete: **paquete híbrido**



Flujo Híbrido

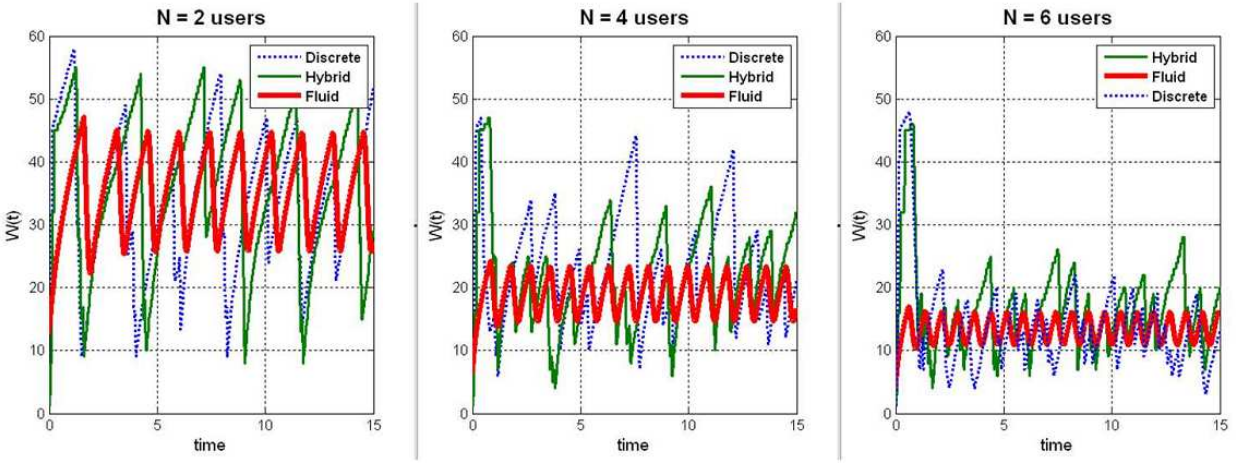
- **Solución:**
  Flujo Híbrido Discreto+Continuo
  1 Flujo discreto "de testigo"

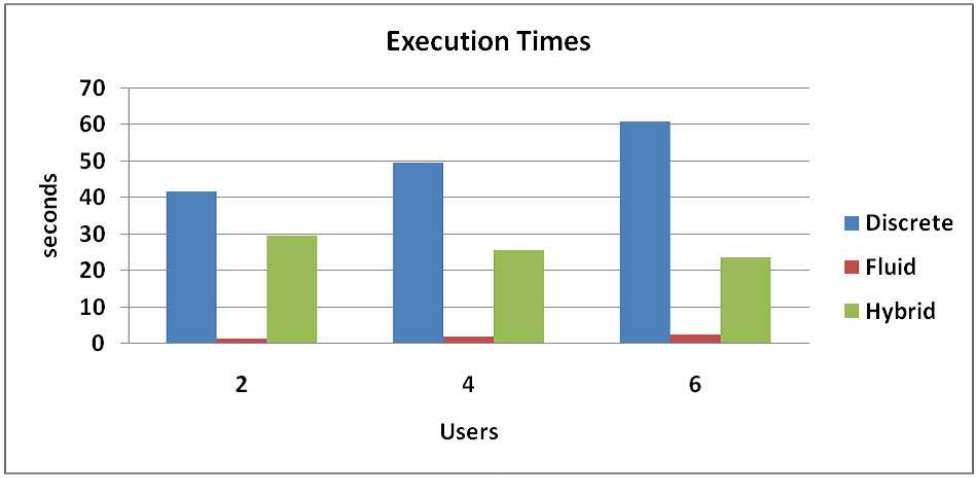
- **Muy sencillo de modelar visualmente**

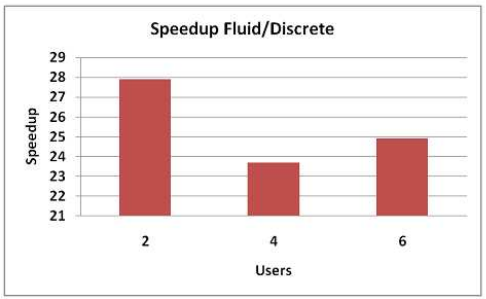


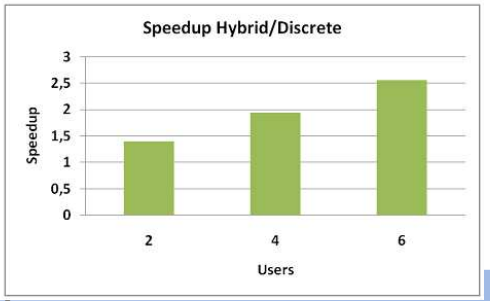
Flujo Híbrido. Resultados.

- Cualitativamente muy aceptable
- Preserva el nivel de detalle del flujo discreto
- Mejora de tiempos respecto del modelo discreto puro
  - 1.5, 2 y 2.5 veces para N=2, 4 y 6 usuarios.





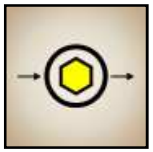


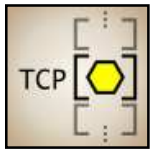


Biblioteca de M&S de Redes

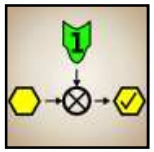
- Nueva biblioteca de modelos de red en PowerDEVS


Emisor

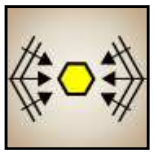

Servidor

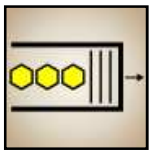


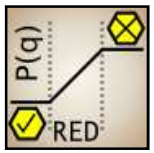
Control de Congestión TCP


Token Bucket

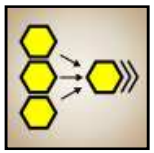

Combinador de Flujo Híbrido

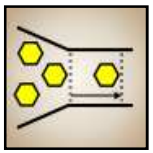

Receptor

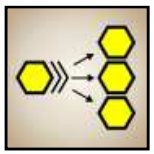

Cola



Random Early Detection


Multiplexor


Canal de Comunicación


Demultiplexor

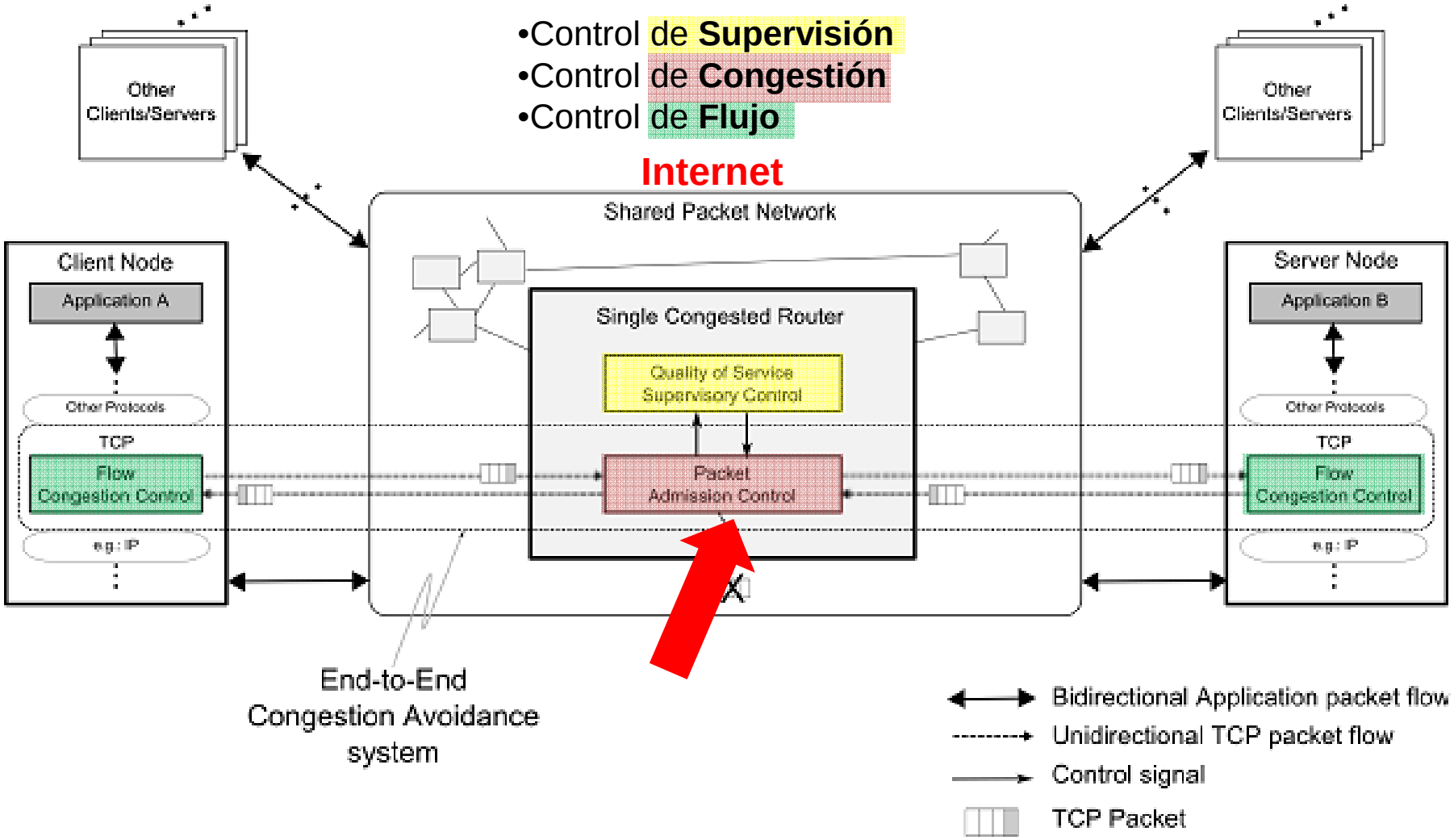
Incluye las nuevas bibliotecas de bajo nivel:

`stdevstool.h` Distribuciones de probabilidad

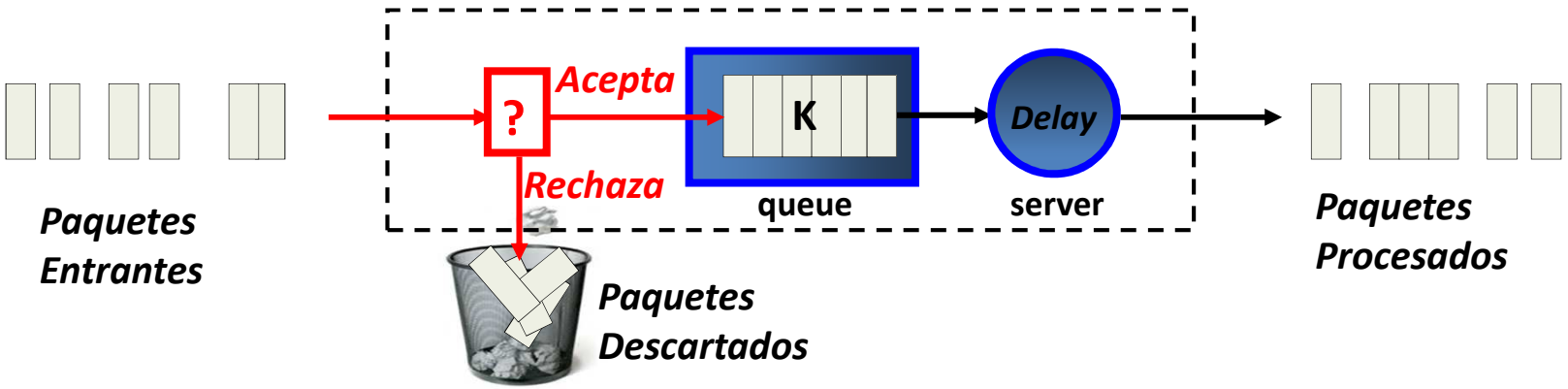
`packettool.h` Estructuras de paquetes multiprotocolos

- Metodología para aplicar Teoría de Control
  - **Control de Admisión en un Router**

- Metodología para aplicar Teoría de Control
  - **Control de Admisión en un Router**



Idea básica

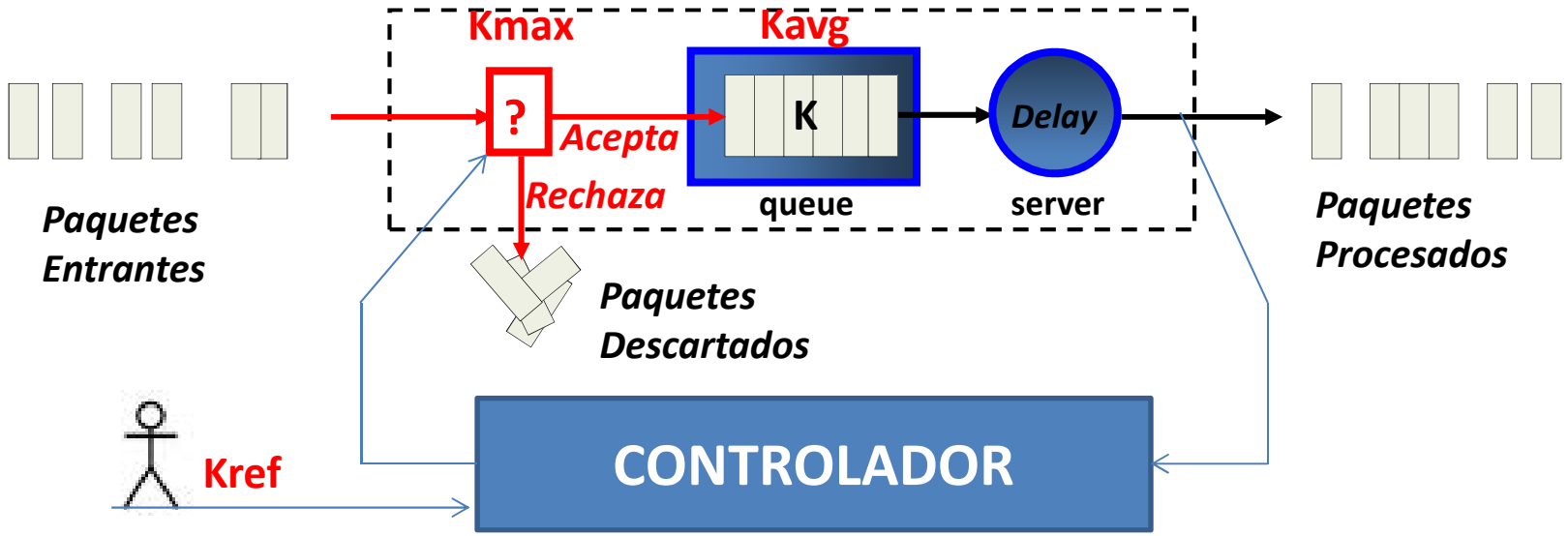


- Router: Capacidad limitada. Muchos flujos compiten entre sí.
  - **Un Requerimientos de Calidad de Servicio:** Longitud Promedio Cola **Kavg**
- Tiempos de arribo y de procesamiento: estocásticos
  - Encolamiento creciente ➔ saturan los buffers ➔ Se incrementan los Retardos ➔ **Se degrada la Calidad de Servicio**

Técnica: **Control de Admisión**
- **Rechazo inteligente** de paquetes para lograr **satisfacer los requerimientos**

Problema: Las técnicas **ad-hoc** son difíciles de analizar matemáticamente:
- **Estabiliad ? Tiempo de Convergencia ? Robustez ? ...**

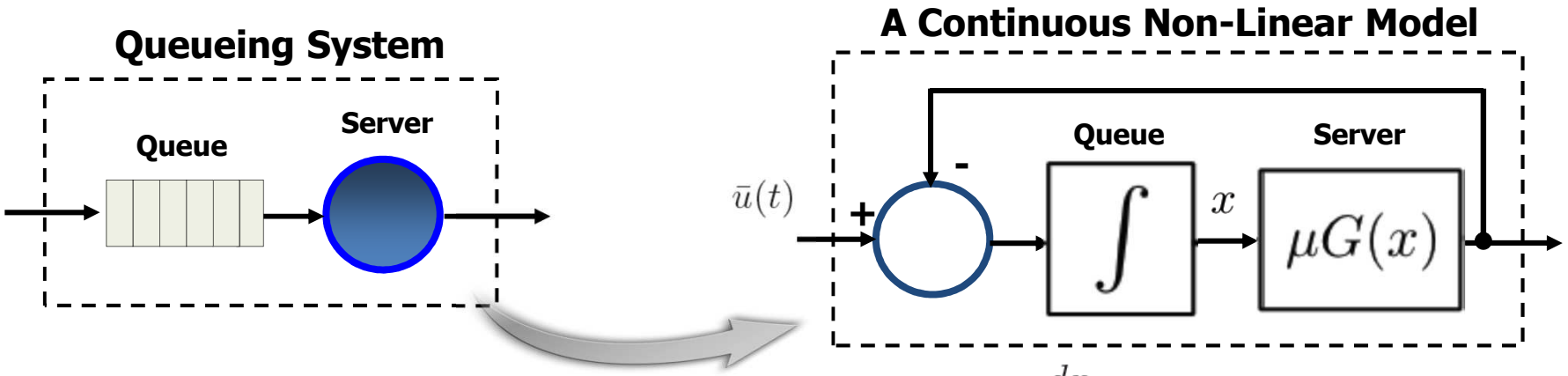
Enfoque con Teoría de Control



- **Objetivo de Control:** Mantener el número **promedio de paquetes encolados** **Kavg** en cierto valor de referencia deseado **Kref**.
- **Formulación mediante Teoría de Control:**
  - **Entrada a la Planta:** Nivel **Kmax** al cual se debe comenzar a descartar paquetes.
  - **Salida de la Planta:** Nivel **Kavg** (en una ventana de tiempo)
  - **Objetivo de Control:** Mantener **Kavg entorno a Kref**
  - **Perturbación:** Características estocásticas del proceso de arribo y de servicio

Modelo continuo

- **Teoría de Control** ➔ herramienta de diseño matemáticamente robusta
  - La **planta** es la **red**
  - **Objetivo de Control**: Satisfacer requerimientos de Calidad de Servicio
- Necesitamos un **modelo** (continuo o discreto) **de la red**
  - **Problema:** Un sistema cola-servidor opera intrínsecamente a **Eventos Discretos**
  - **Solución:** Aproximar la red mediante un **modelo Continuo No Lineal**

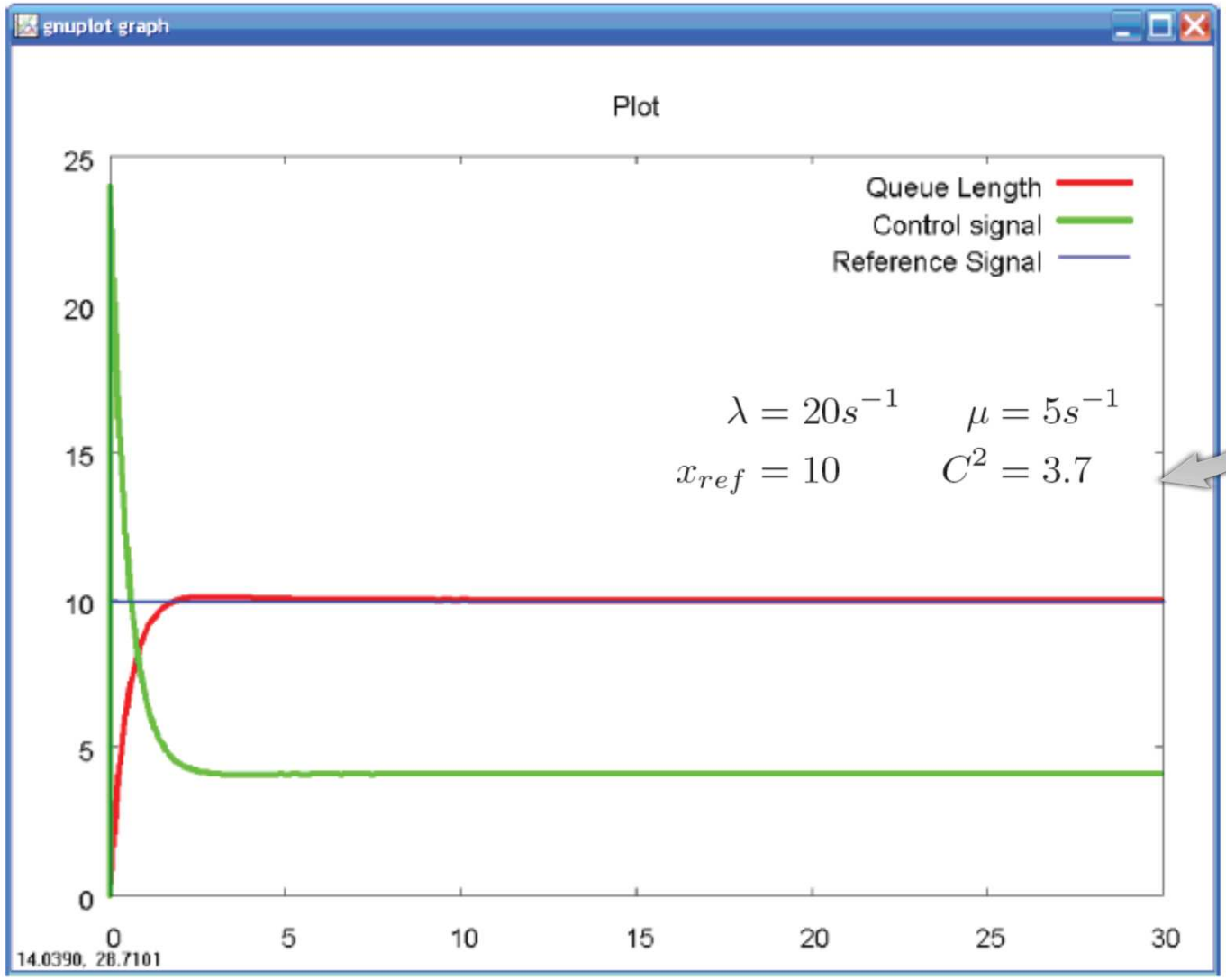


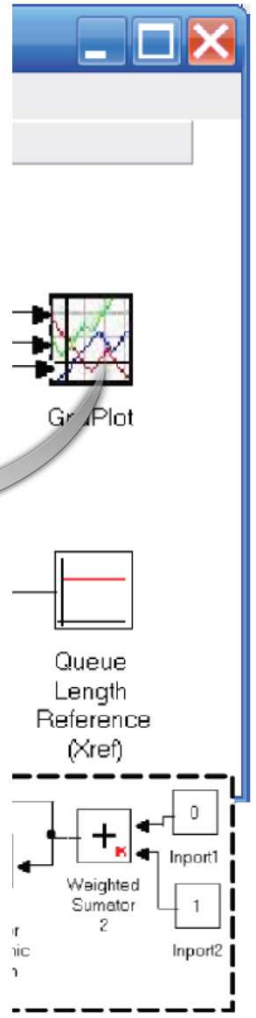
Fluid Flow Non-linear approximation
method for an M/G/1 *system*
(Tipper and Sundareshan, 1990)

$$\frac{dx}{dt} = \bar{u}(t) - \mu G(x(t))$$

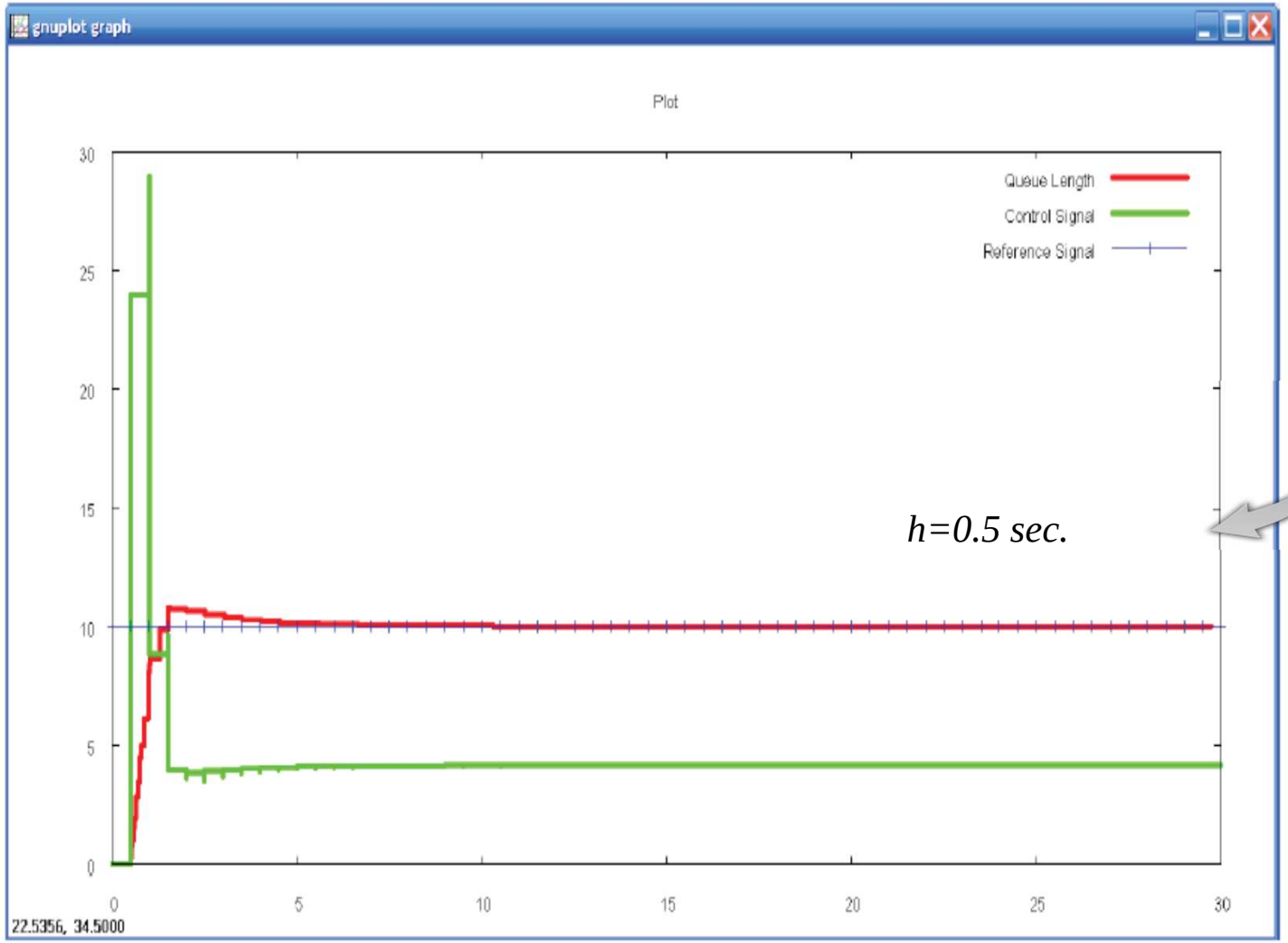
$$G(x(t)) = \frac{x(t) + 1 - \sqrt{x^2(t) + 2C^2x(t) + 1}}{1 - C^2}$$

- Modelo de la Planta:
  - Apr
    line
    col
- Contr
  - Dis
    de
- Actua
  - Un
    reg
    tip
- Herra
  - Imp
    QS
    nur
    cua

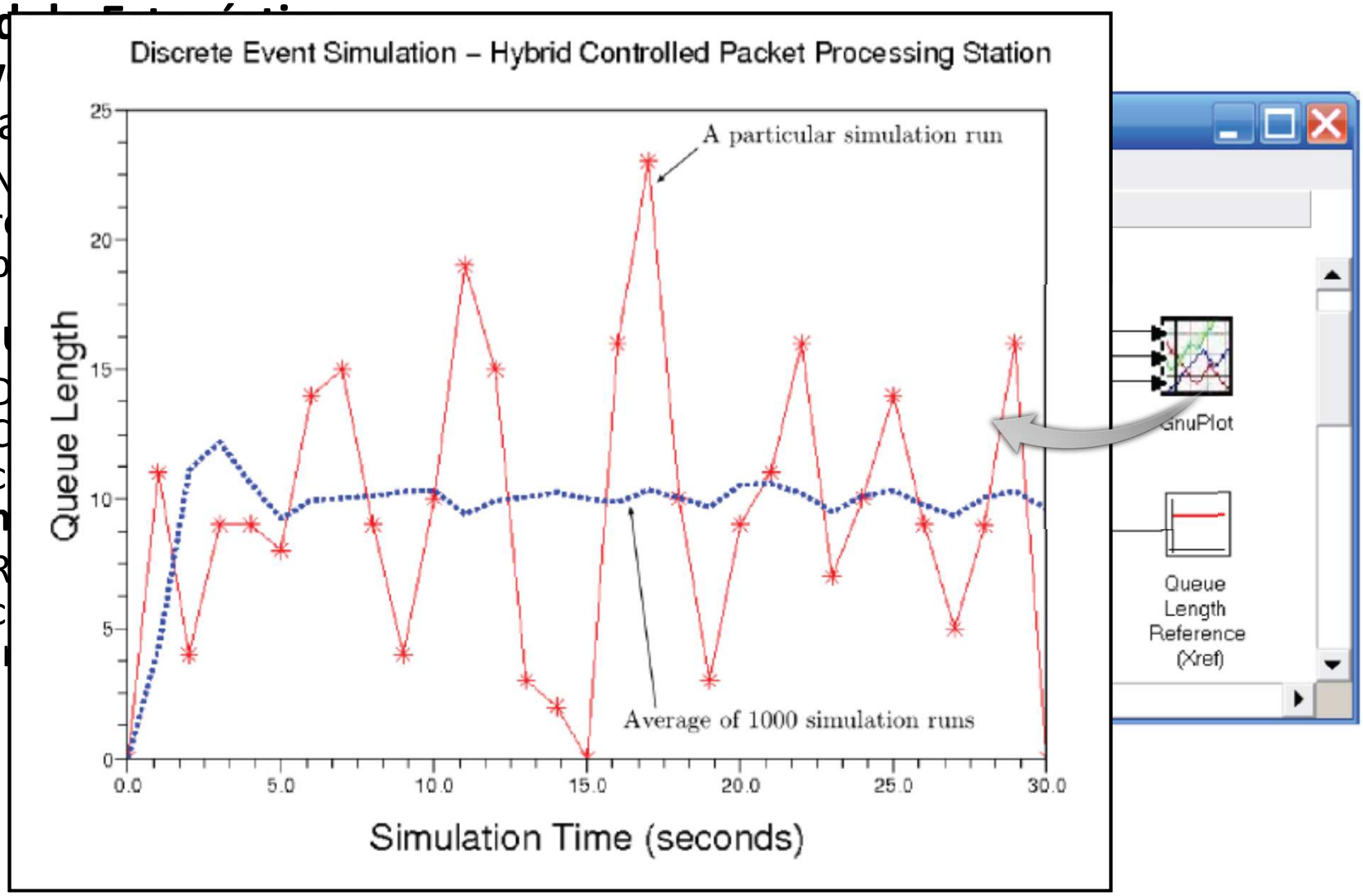




- **Co**
  dis
  (Fc
- Tie
  –
  –
- Re
  de
  de
  –

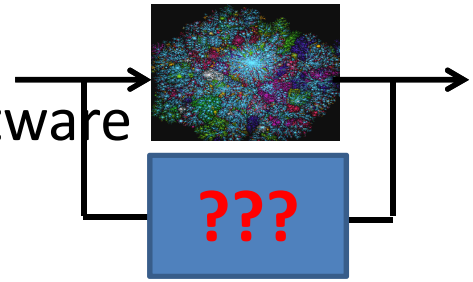


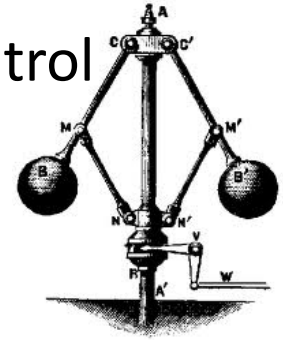
- Control PI Discretizado
- **Mod**
  a Ev
  de la
  - N
    r
    p
- Requ
  - D
    C
    c
    m
  - R
    c
    (**r**



Conclusiones Parciales

- Metodología en capas
  - Desarrollo en paralelo de
    - herramientas genéricas
    - para atacar problemas particulares
- Coexistencia de Controladores heterogéneos
  - DEVS permite el modelado simultáneo de subsistemas heterogéneos
  - Promueve la interdisciplina reduciendo riesgos (es decir, costos)
  - Puente entre la Teoría Clásica de Control y la joven disciplina de Control de Sistemas de Software

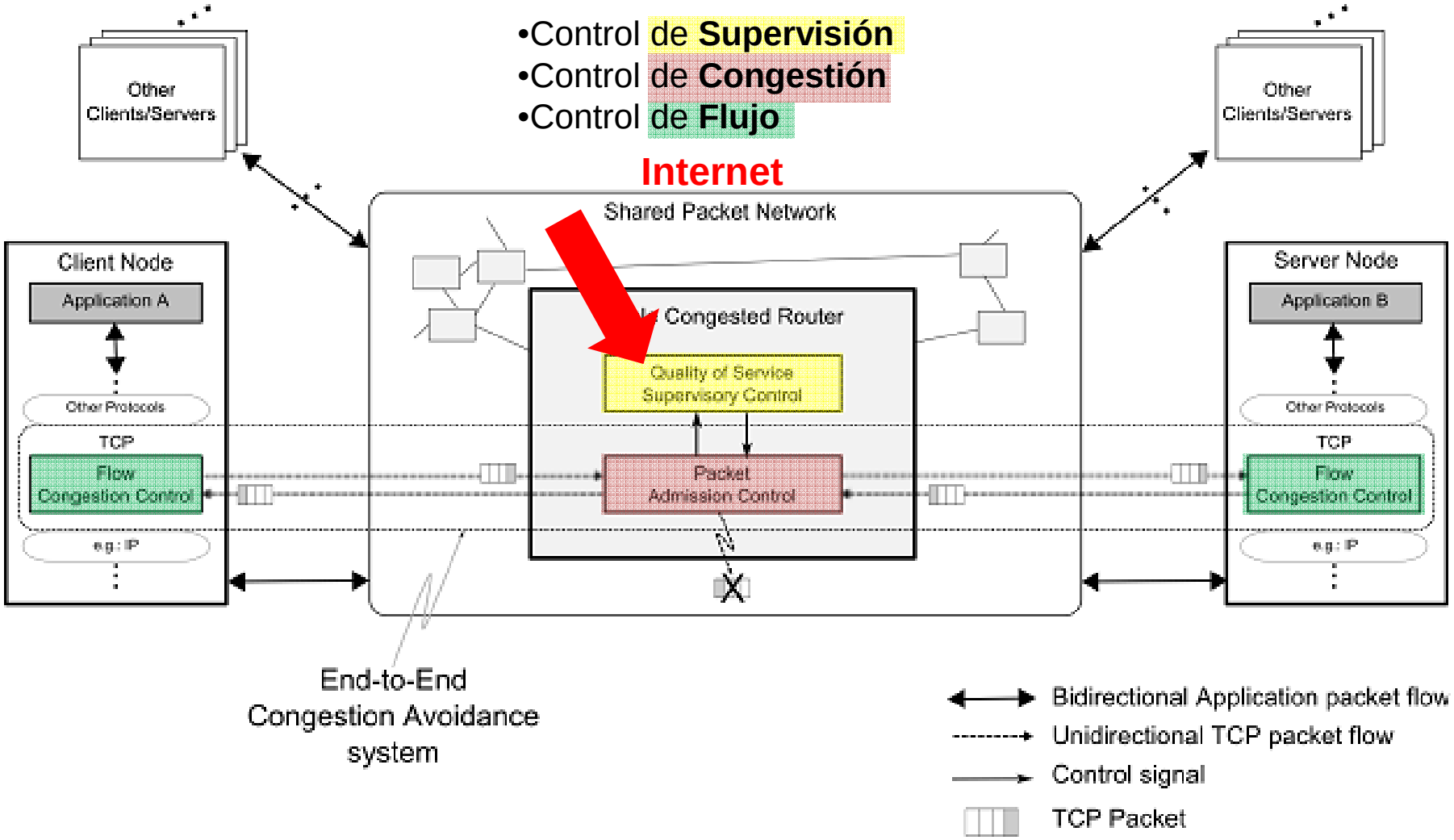




Bonus ! Parte 3)

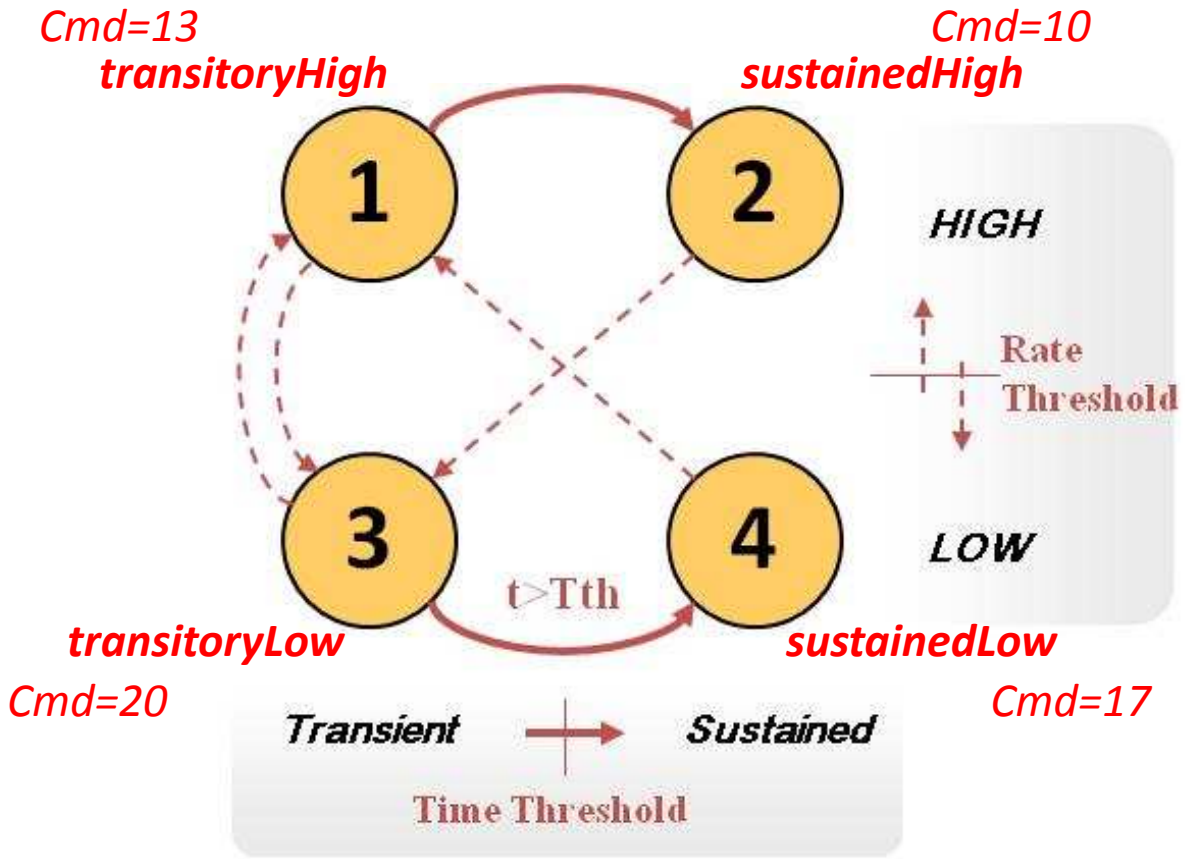
- Implementación directa de Controladores en Real Time
  - **Control Supervisorio basado en Medición de Tasa**

- Implementación directa de Controladores en RT
  - **Control Supervisorio basado en Medición de Tasa**

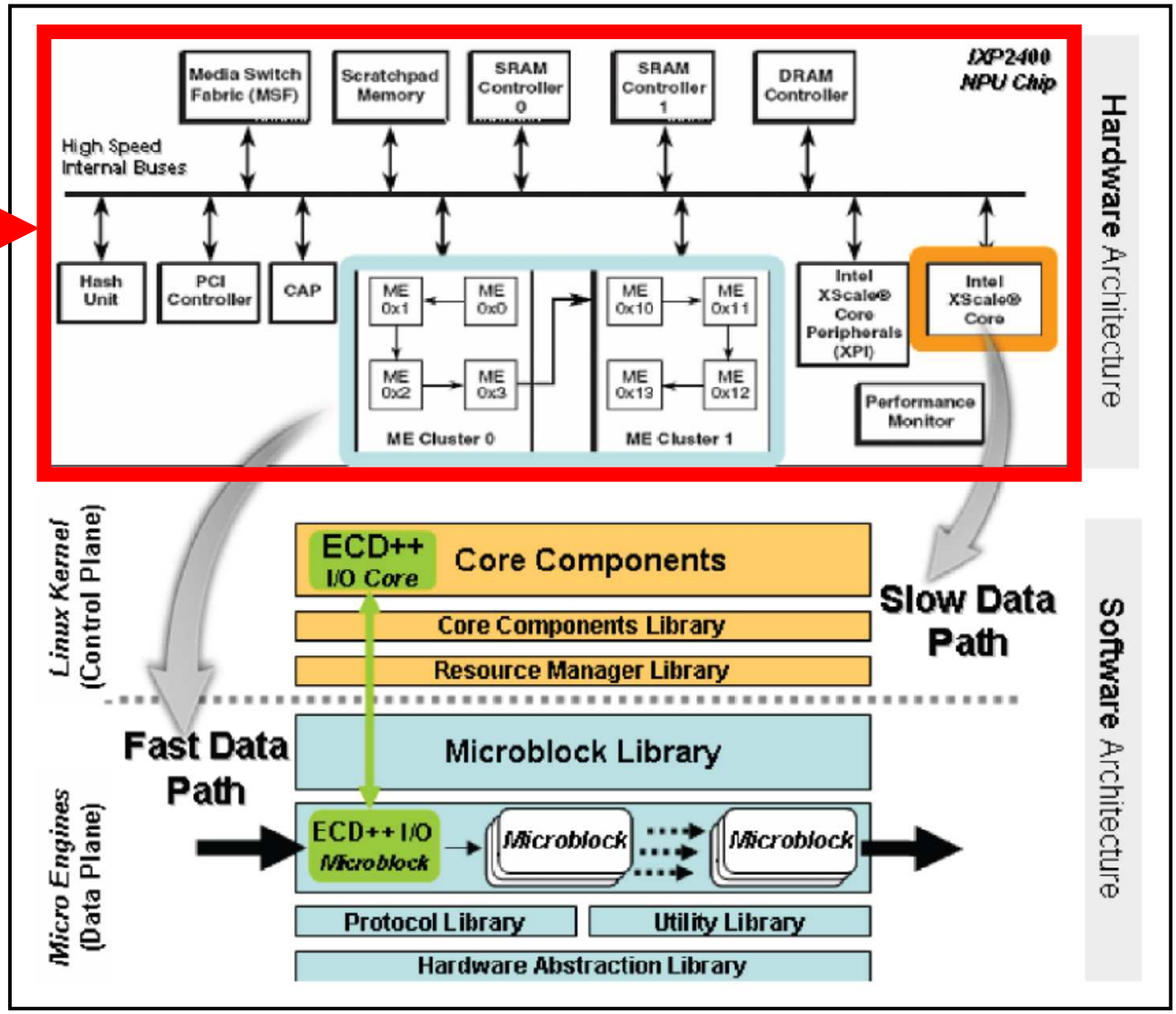


- Logramos un proceso unificado de desarrollo sin cambios de formalismos
- Supongamos que **ya tenemos un modelo satisfactorio** de un controlador discreto
- Queremos utilizar al **modelo como el propio producto final**, sin recodificar para plataformas específicas

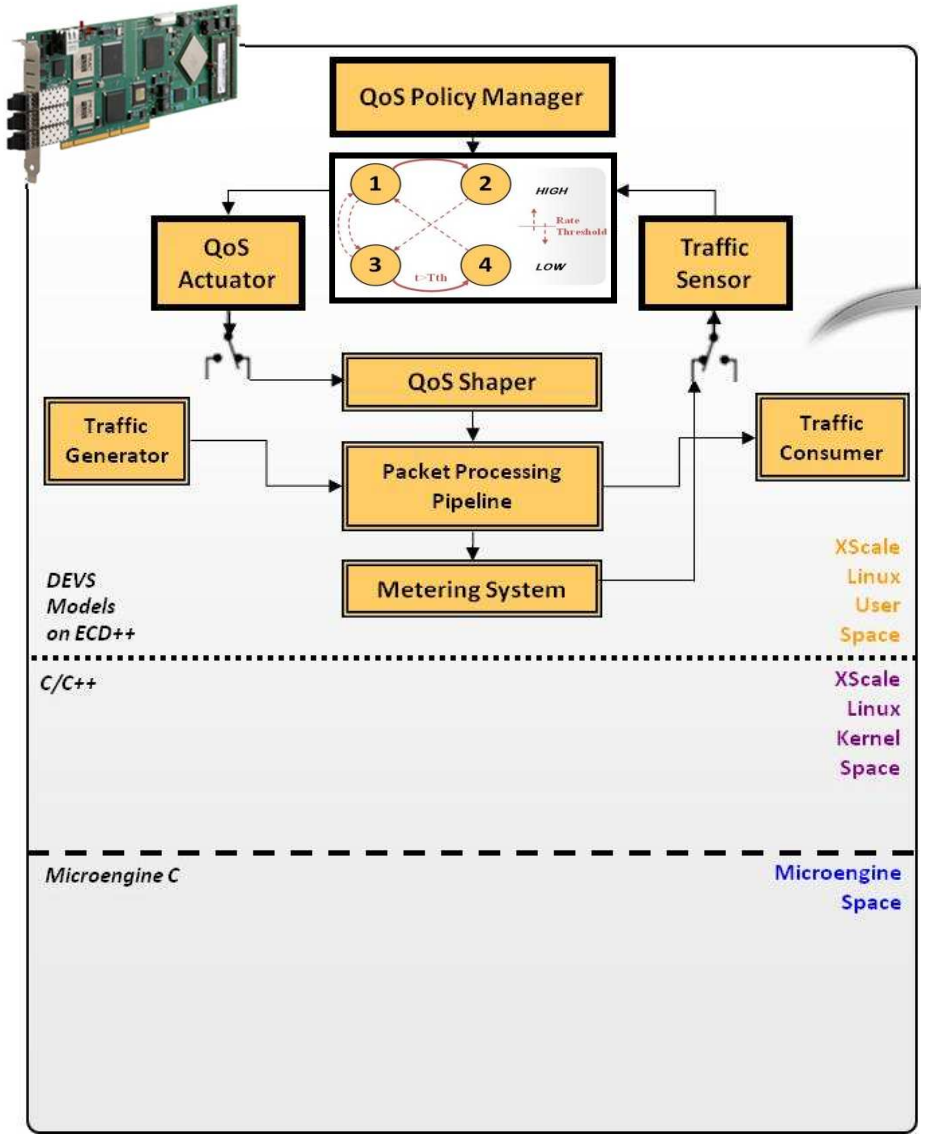
- Máquina de estados finitos
  - **Manejo de Longitud de Cola basado en Medición de Tasa**



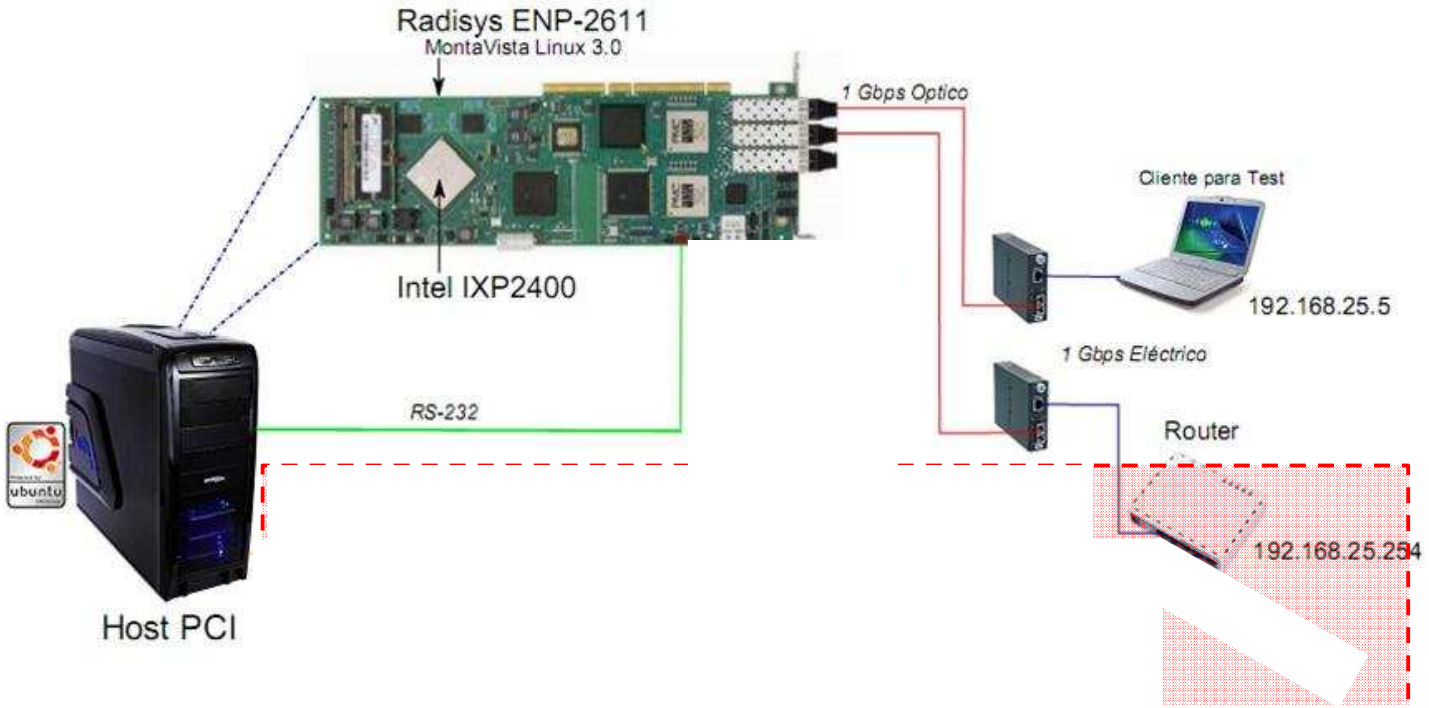
DEVS embebido en Tiempo Real



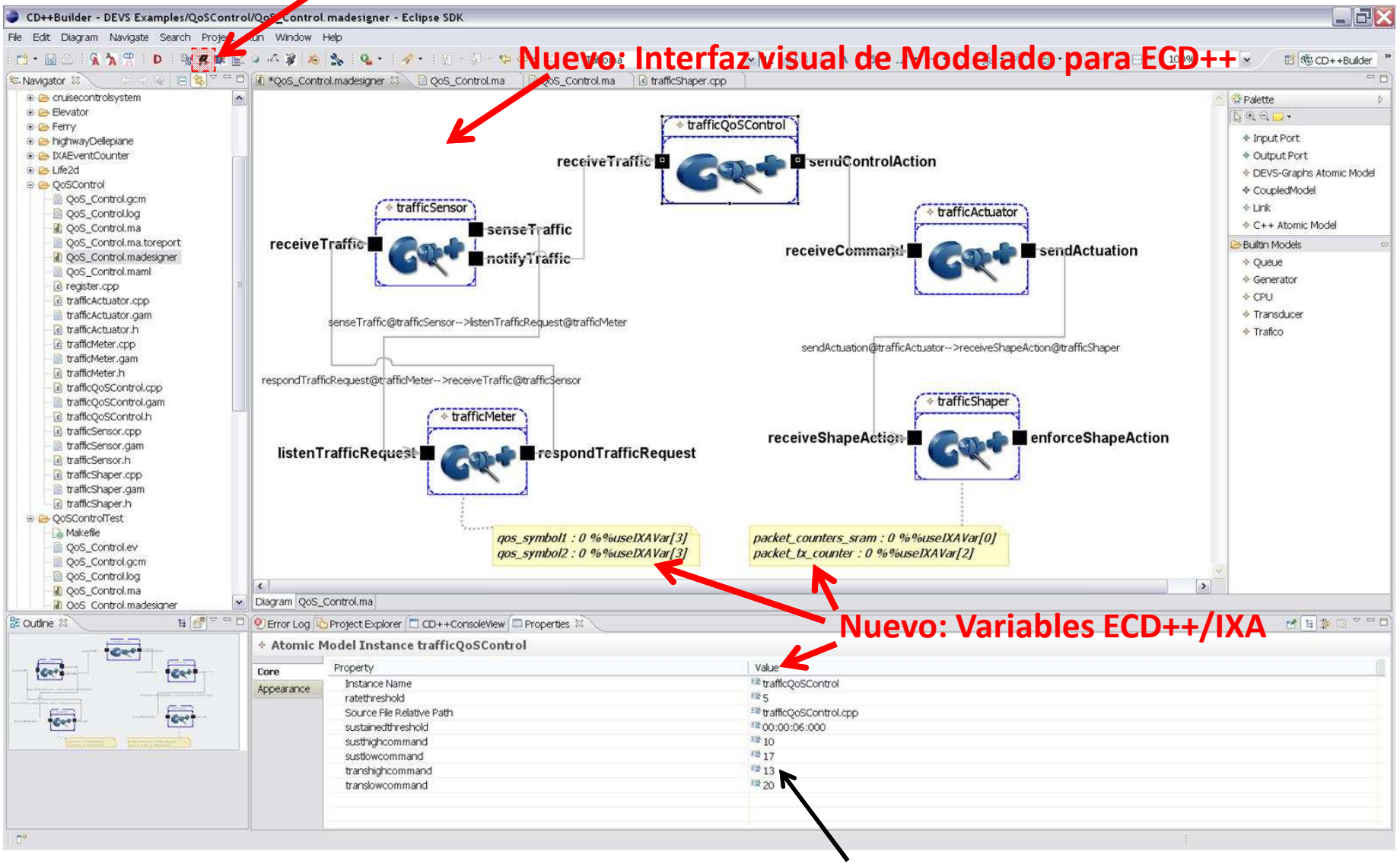
DEVS embebido en Tiempo Real



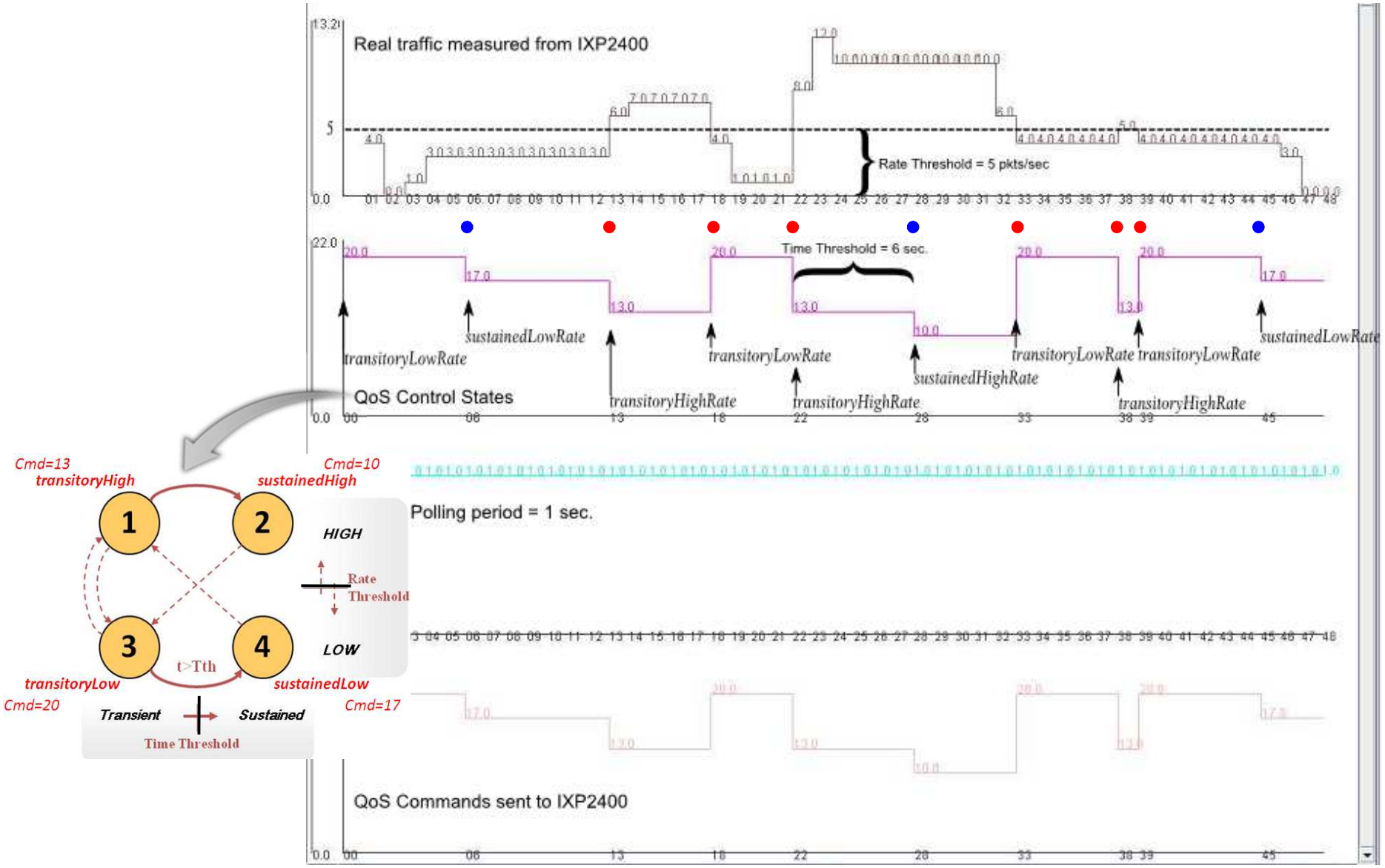
Simulación de Modelos DEVS Embebida en Tiempo Real


Radisys ENP-2611
MontaVista Linux 3.0
1 Gbps Optico
Cliente para Test
192.168.25.5
Intel IXP2400
1 Gbps Eléctrico
RS-232
Router
192.168.25.254
Host PCI

Nuevo: Generación automática de código para ECD++/IXA



Resultados con tráfico real



Más conclusiones

- **Diseño de controladores embebidos** basado en el formalismo DEVS
- Implementación de un **simulador DEVS en un Procesador de Red**
  - ECD++
  - Intel IXP2400
- **El simulador interactúa con el hardware** de manipulación de paquetes
  - Controla tráfico en Tiempo Real, a modo Hardware-in-the-Loop (HIL)
- **Ciertos modelos son reemplazados** por sus contrapartes reales en hardware
  - cuando el simulador pasa operar en modo HIL
- **Otros modelos permanecen intactos**
  - adoptando el rol del **producto final**

Preguntas

Gracias !

rcastro@dc.uba.ar