



Sistemas Auto-Adaptables

Gabriel Moreno

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Copyright 2019 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

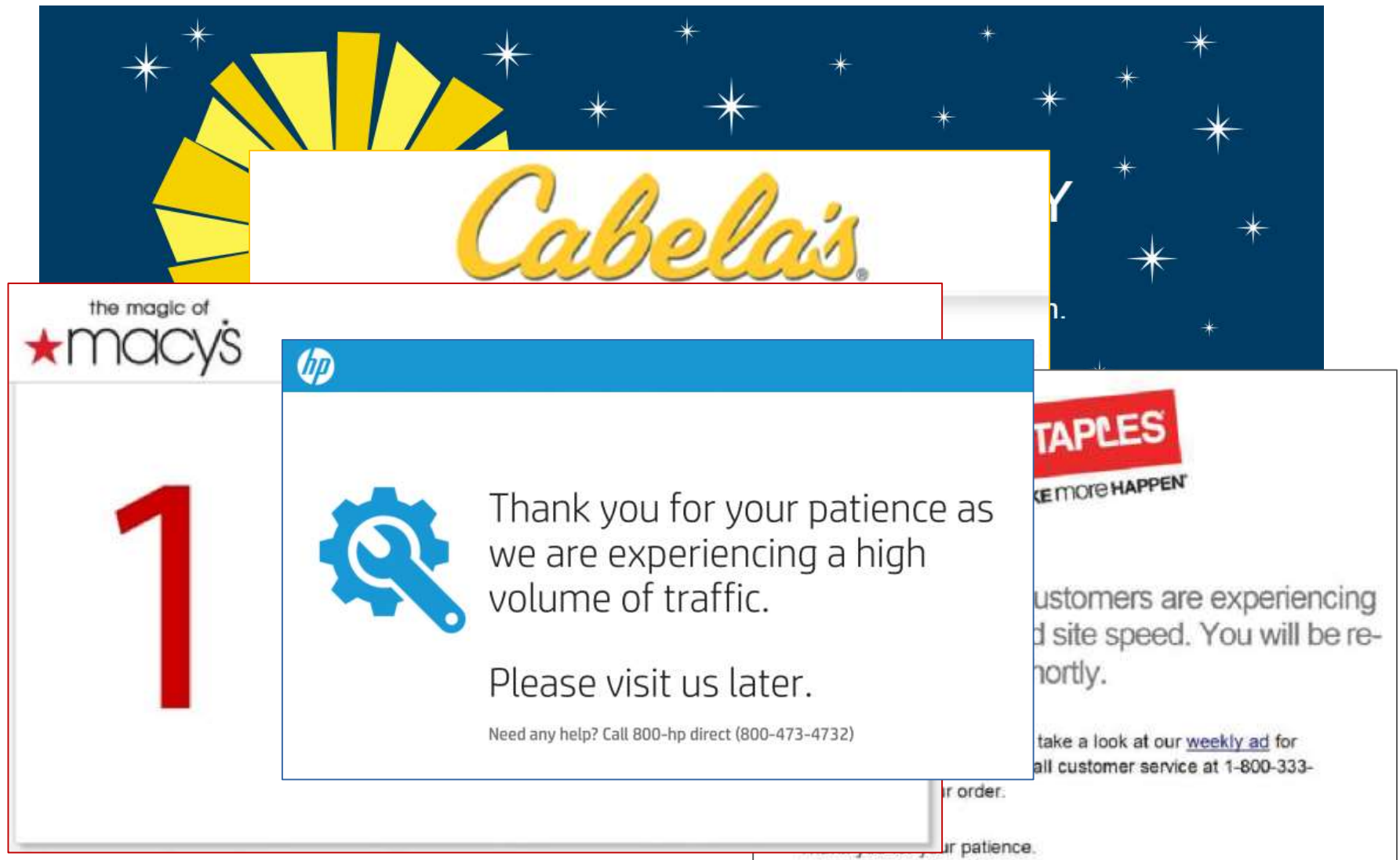
NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM19-0661

Black Friday



Un sistema auto-adaptable es un sistema capaz de cambiar su comportamiento y estructura para adaptarse sin intervención humana a los cambios en sí mismo y su entorno de operación

R. de Lemos et al. "Software engineering for self-adaptive systems: A second research roadmap." *Software Engineering for Self-Adaptive Systems II. Lecture Notes in Computer Science*, 2013

Características de Auto-Adaptación

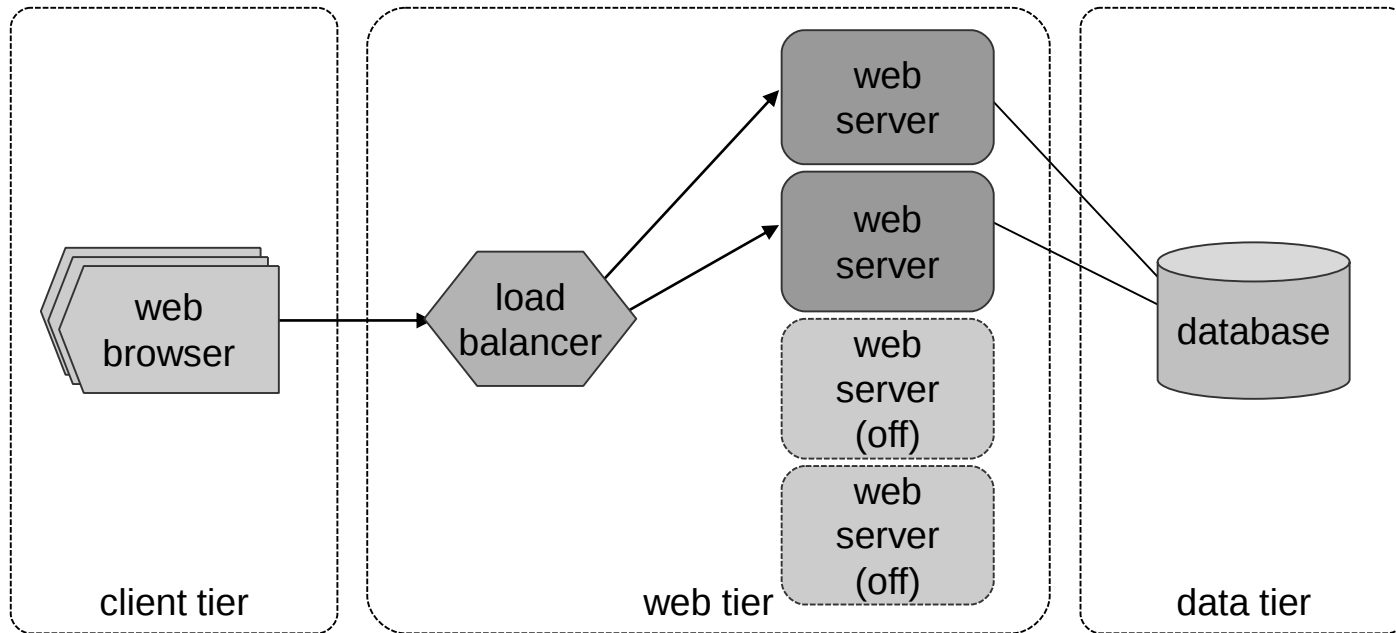
Auto-configuración

Auto-reparación

Auto-optimización

Auto-protección

Ejemplo: Aplicación Web 3-tier RUBiS



Entorno:

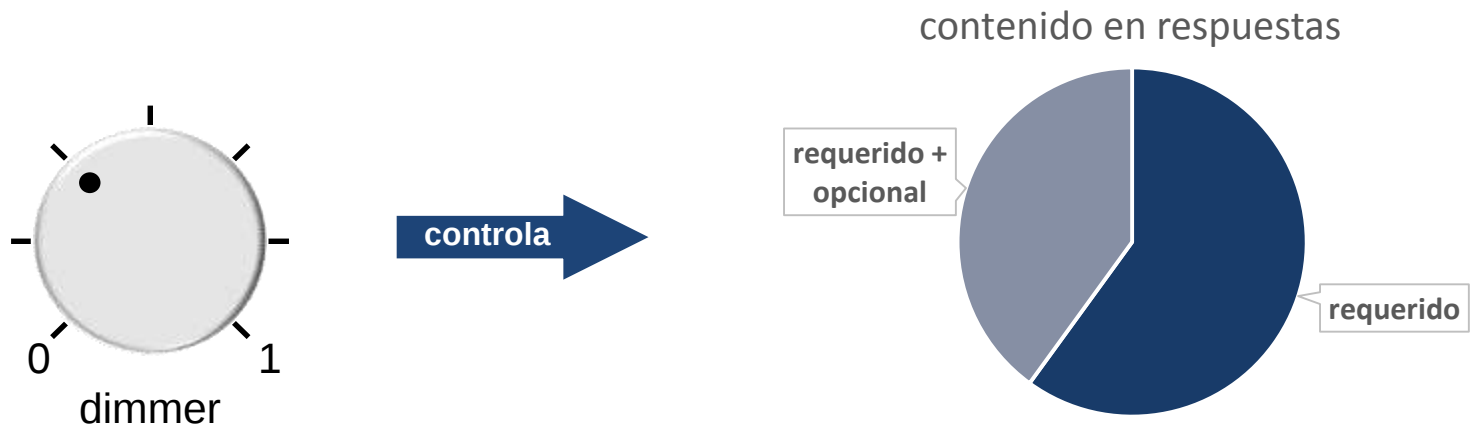
- Tráfico al website (variable)

Objetivo:

- Mantener tiempo de respuesta bajo un umbral
- Servir la mayor cantidad de contenido opcional
- Minimizar el costo de operación (nro. de servidores)

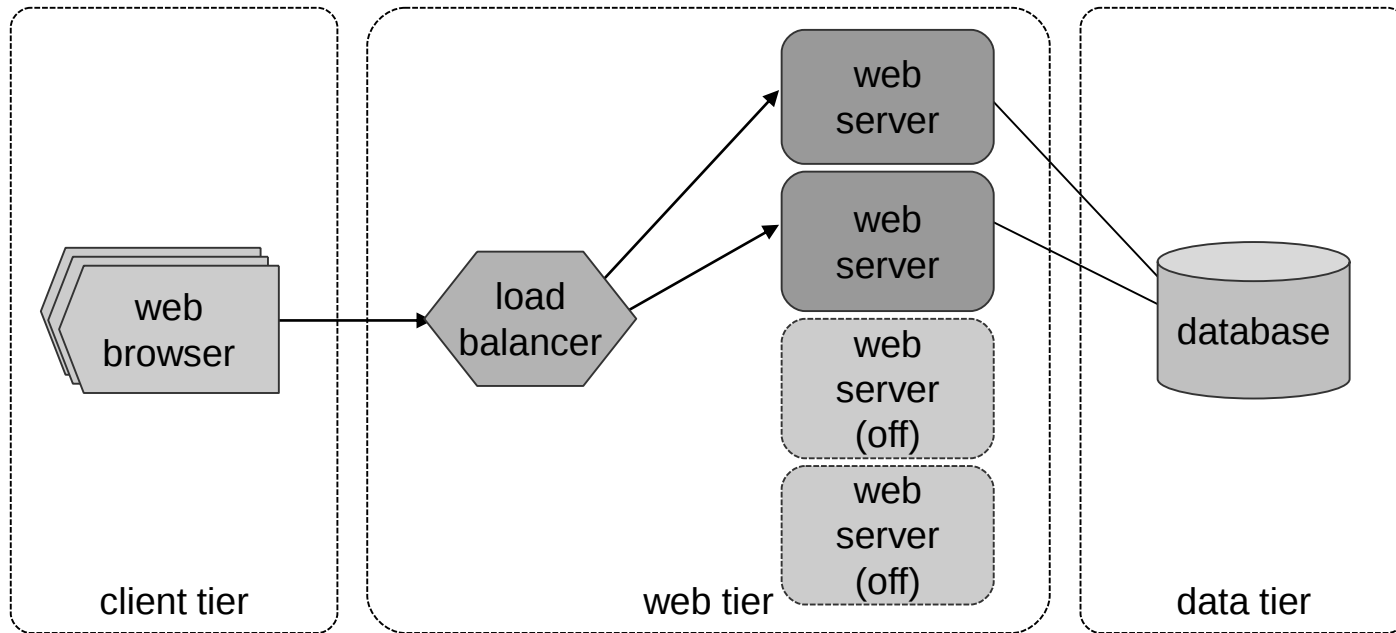
RUBiS con Brownout

Brownout: método para regular la carga en el sistema inducida por el tráfico de usuarios.



C. Klein et al. "Brownout: building more robust cloud applications." In *Proceedings of the International Conference on Software Engineering*, 2014

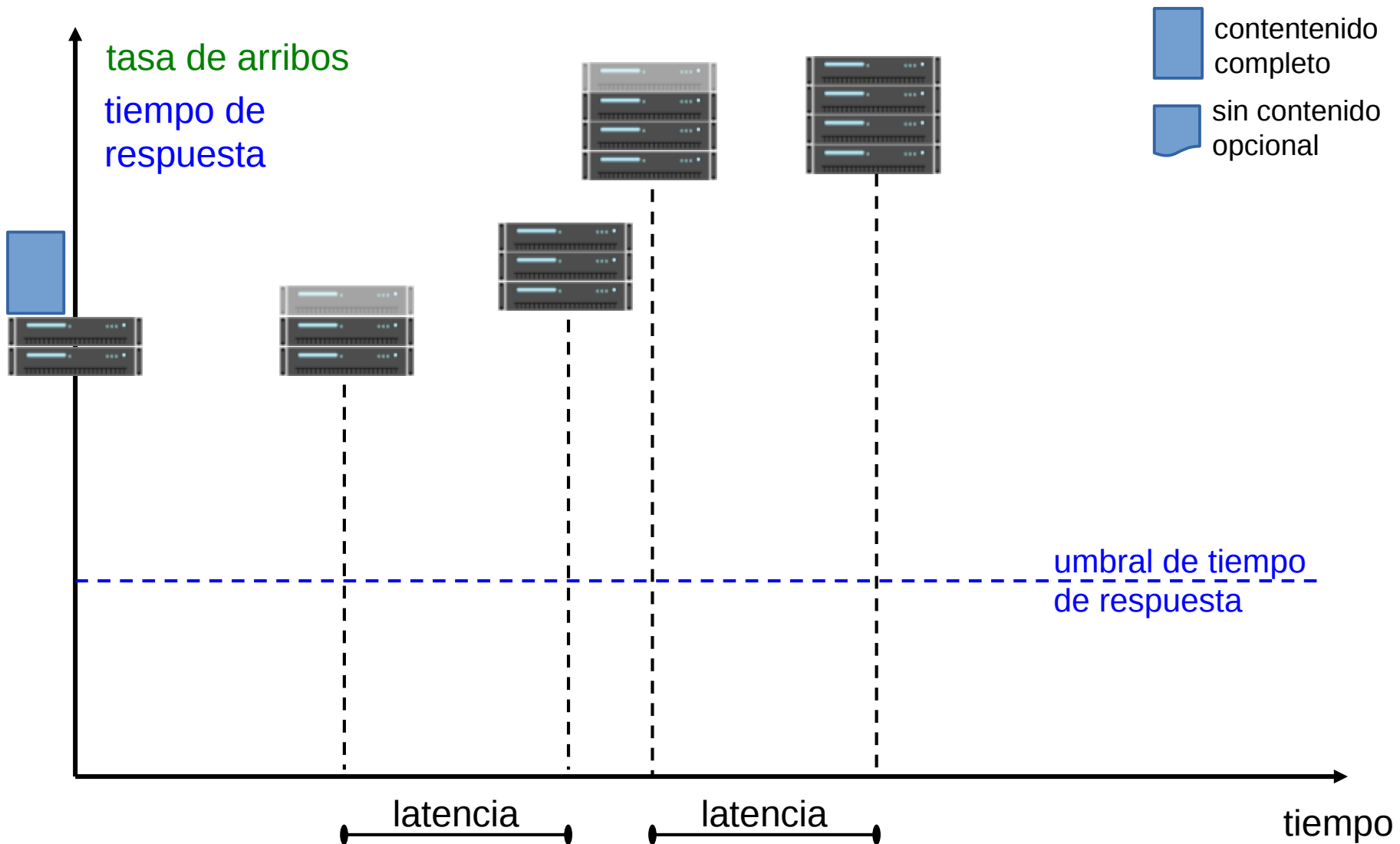
Ejemplo: Aplicación Web 3-tier RUBiS



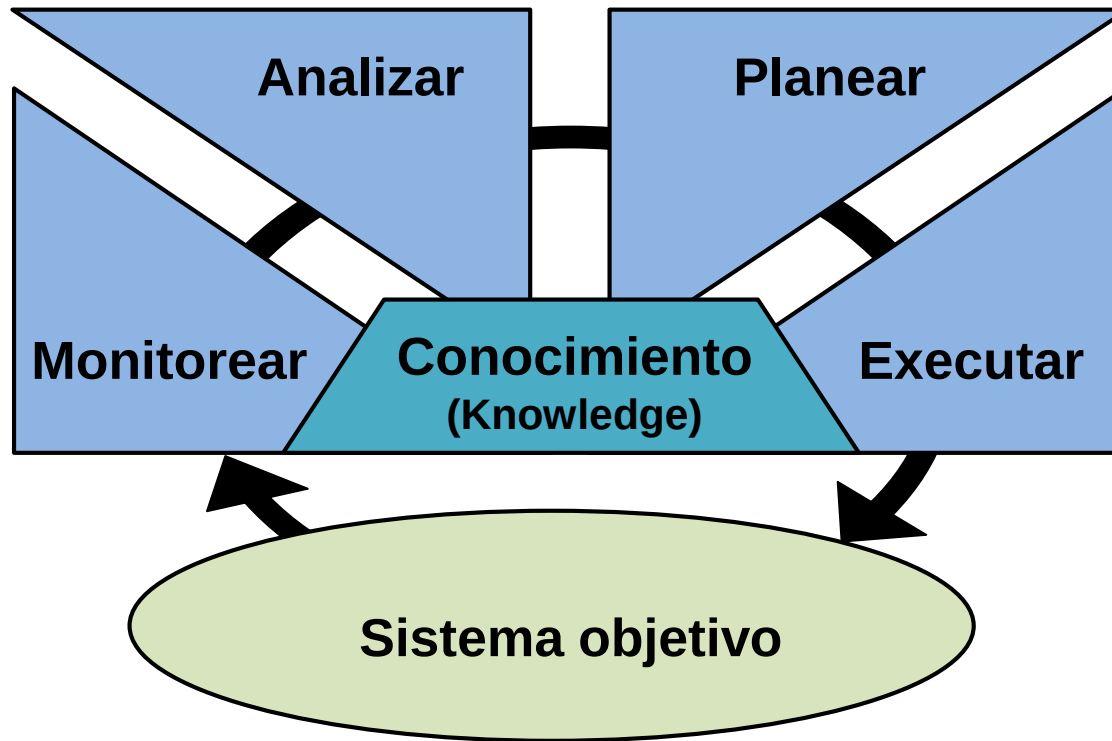
Acciones de adaptación:

- Agregar/sacar servidor
- Incrementar/reducir proporción de contenido opcional

Ejemplo de Adaptación Reactiva



MAPE-K



J.O. Kephart, and D.M. Chess. "The vision of autonomic computing."
Computer 36, no. 1, 2003

Realizando la Visión

Monitoreo

- Mantener modelos del sistema

Análisis

- Análisis de modelos

Planificación

- Reglas
- Inteligencia Artificial

Ejecución

- Cosas modificables por software
 - Tecnologías habilitantes: Cloud, SDN, IaC
- Humano en el ciclo

Reglas Evento-Condición-Acción (ECA)

Método básico para análisis y planeación

Creadas manualmente en base a heurísticas

<i>Cuando la utilización promedio de CPUs pasa el 60%,</i>	← evento
<i>si no se ha alcanzado el límite de servidores,</i>	← condición
<i>agregar un servidor</i>	← acción

Ejemplo: Amazon EC2 Auto-scaling

Ventajas/Desventajas de Reglas ECA

Ventajas

- Cada regla individualmente es fácil de entender
- Las reglas se pueden ir agregando incrementalmente

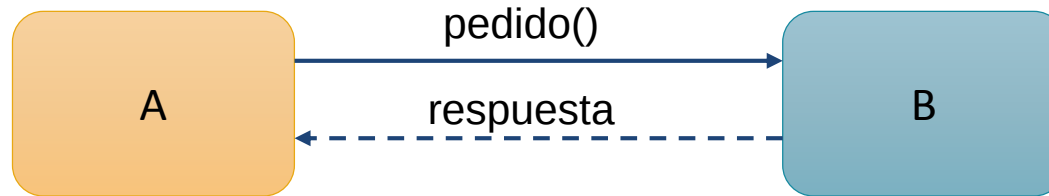
Desventajas

- El manejo de un objetivo puede estar esparcido en múltiples reglas, dificultando comprensión
- Múltiples reglas pueden interactuar y tener conflictos
- Es difícil garantizar la cobertura de todo el espacio de adaptación

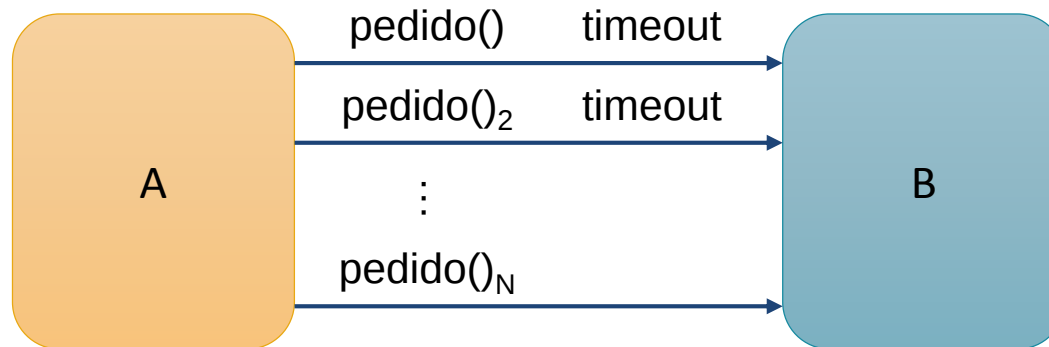
H. Ghanbari, B. Simmons, M. Litoiu, & G. Iszlai. "Exploring alternative approaches to implement an elasticity policy." *Proceedings of the International Conference on Cloud Computing*, 2011

Límites de Adaptación de Bajo Nivel

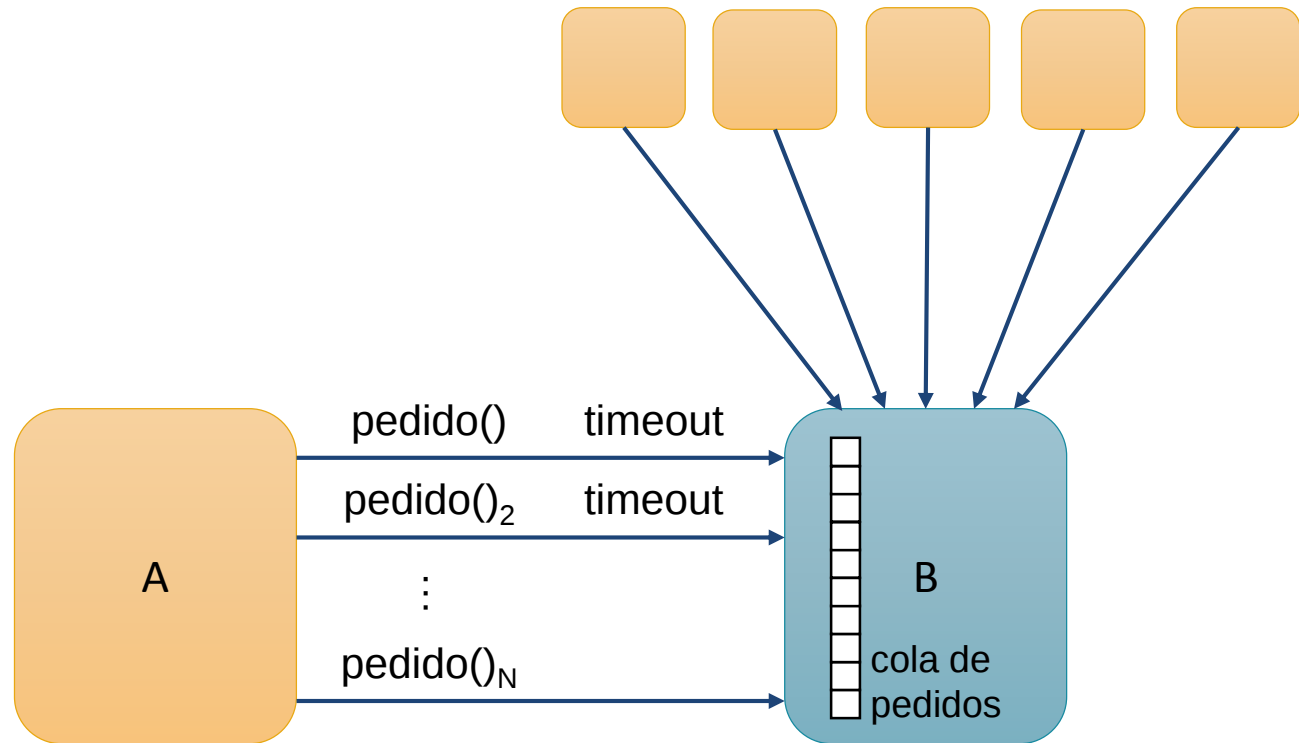
Ejemplo: componente A invoca servicio en componente remoto B (caso normal)



Falla (vista local)



Límites de Adaptación de Bajo Nivel



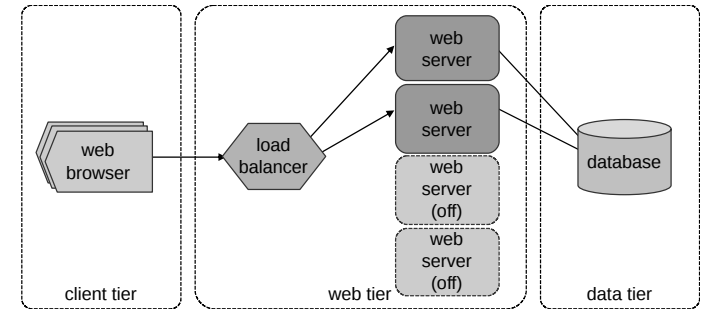
Además:

- Código esparcido, frágil y costoso de mantener

Auto-Adaptación Basada en Arquitectura

La arquitectura del software:

- provee perspectiva global del sistema
- expone propiedades del sistema
 - tiempo respuesta, confiabilidad, etc.
- con el uso de estilos arquitecturales, impone restricciones a los cambios posibles (permite verificar la integridad de la adaptación)



Foco en la estructura dinámica de la arquitectura (componentes y conectores)

Permite usar distintos análisis existentes para arquitecturas

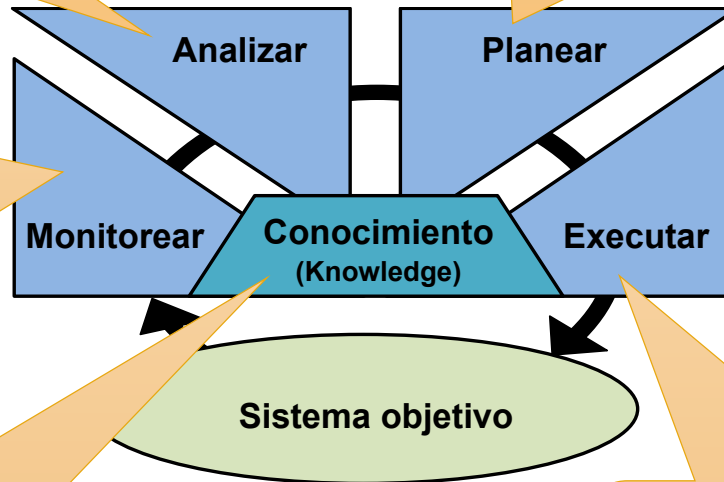
D. Garlan, B. Schmerl, & S.-W. Cheng. "Software architecture-based self-adaptation." In *Autonomic Computing and Networking*, 2009

Auto-Adaptación Basada en Arquitectura

Los objetivos de adaptación normalmente son funciones de la propiedades de la arquitectura

- Estilo arquitectural guía los posibles cambios
- Uso de análisis de arquitectura para evaluar impacto de posibles cambios

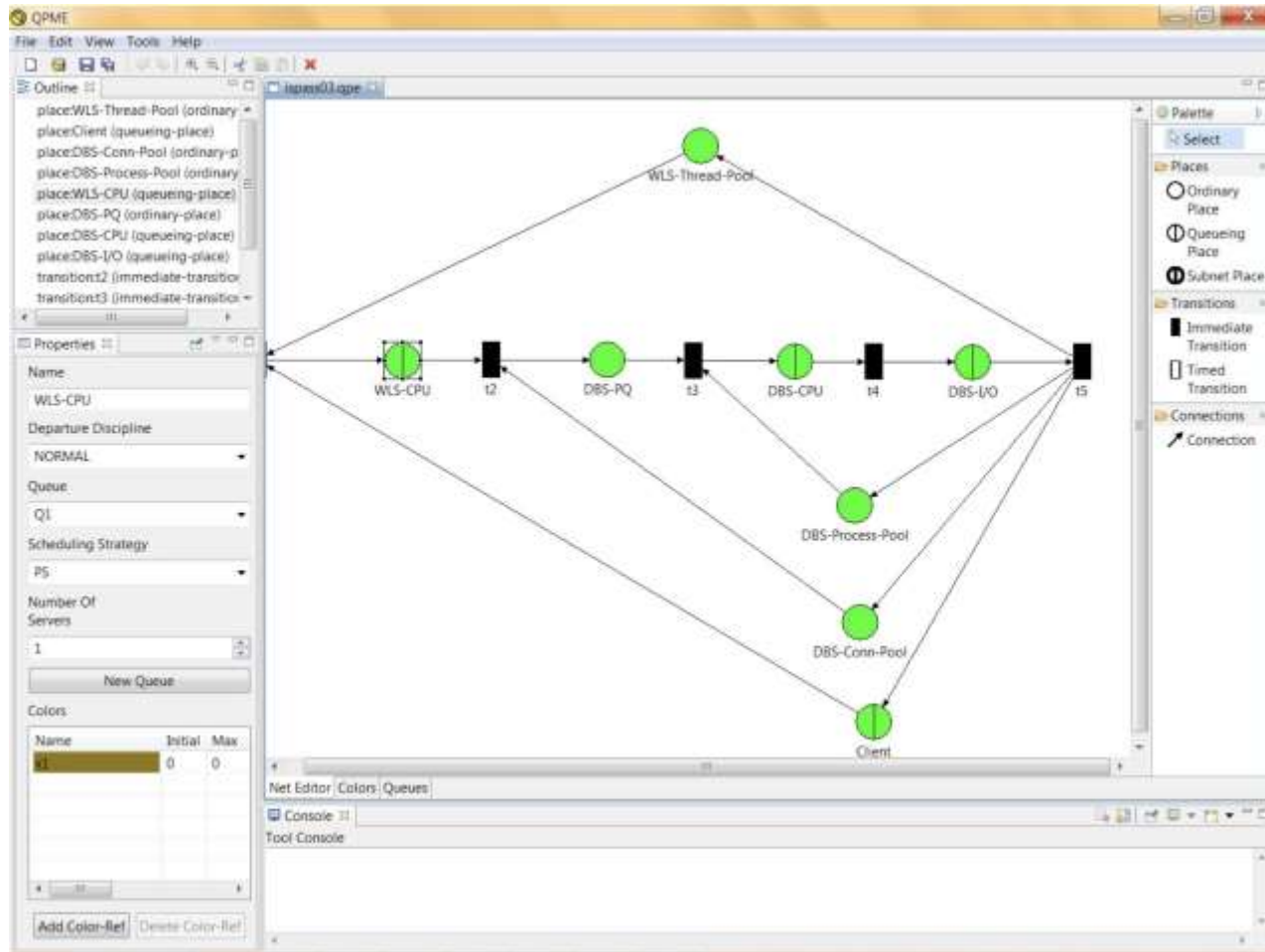
Mantiene actualizado el modelo de la arquitectura (estructura y propiedades)



Modelo de la arquitectura del sistema en un formato procesable por computadora

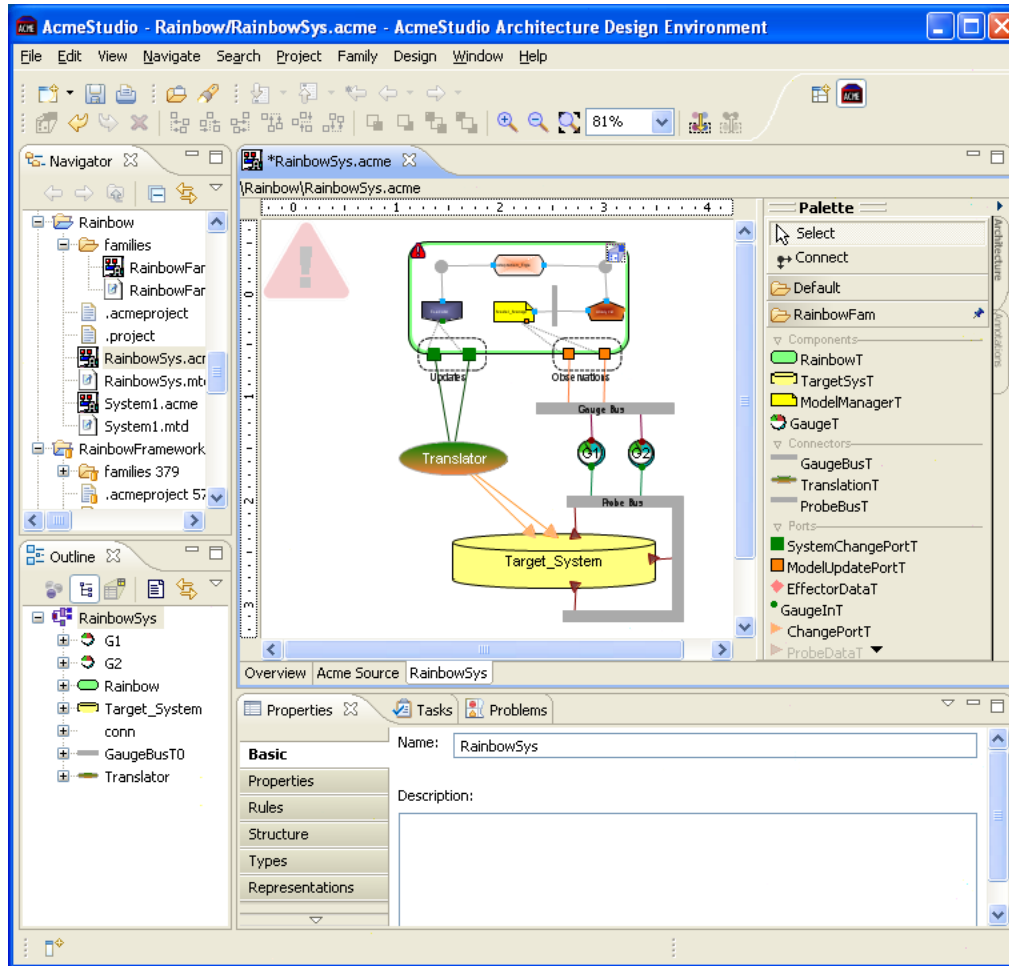
Operadores definidos sobre la arquitectura facilitan expresar los cambios (p.ej., conectar componentes)

Análisis con Modelos—Queuing Petri Nets



S. Kounev, S. Spinner & P. Meier, P. “QPME 2.0-a tool for stochastic modeling and analysis using queueing Petri nets.” In *From active data management to event-based systems and more*, Springer, 2010

Análisis con Modelos—AcmeStudio



B. Schmerl and D Garlan. “AcmeStudio: Supporting Style-Centered Architecture Development.”
In *Proceedings of the International Conference on Software Engineering*, 2004

Tácticas y Estrategias

Táctica: acción de adaptación primitiva que deja el sistema en un estado consistente

- tiene asociada una pre-condición de aplicación

Estrategia: un árbol de decisión construido con tácticas, que tiene como objetivo adaptar el sistema para solucionar un problema específico

Stitch

Lenguaje para auto-adaptación basada en arquitectura

```
1 strategy FastReduceResponseTime [ M.LB0.avgResponseTime > M.RT_THRESHOLD] {
2   t1: (!isDimmerMin) -> TSetMinDimmer() {
3     t1a: (success) -> done;
4     t1b: (default) -> fail;
5   }
6   t2: (isDimmerMin && canAddServer) -> TAddServer() @[180000 /*ms*/] {
7     t1a: (success) -> done;
8     t1b: (default) -> fail;
9   }
10  t3: (default) -> fail;
11 }
```

S.-W. Cheng, & D. Garlan. "Stitch: A language for architecture-based self-adaptation."
Journal of Systems and Software, 2012

Decisiones de Adaptación

En general, más de una opción es aplicable para adaptación.

La decisión consiste en elegir la mejor de acuerdo a un criterio.

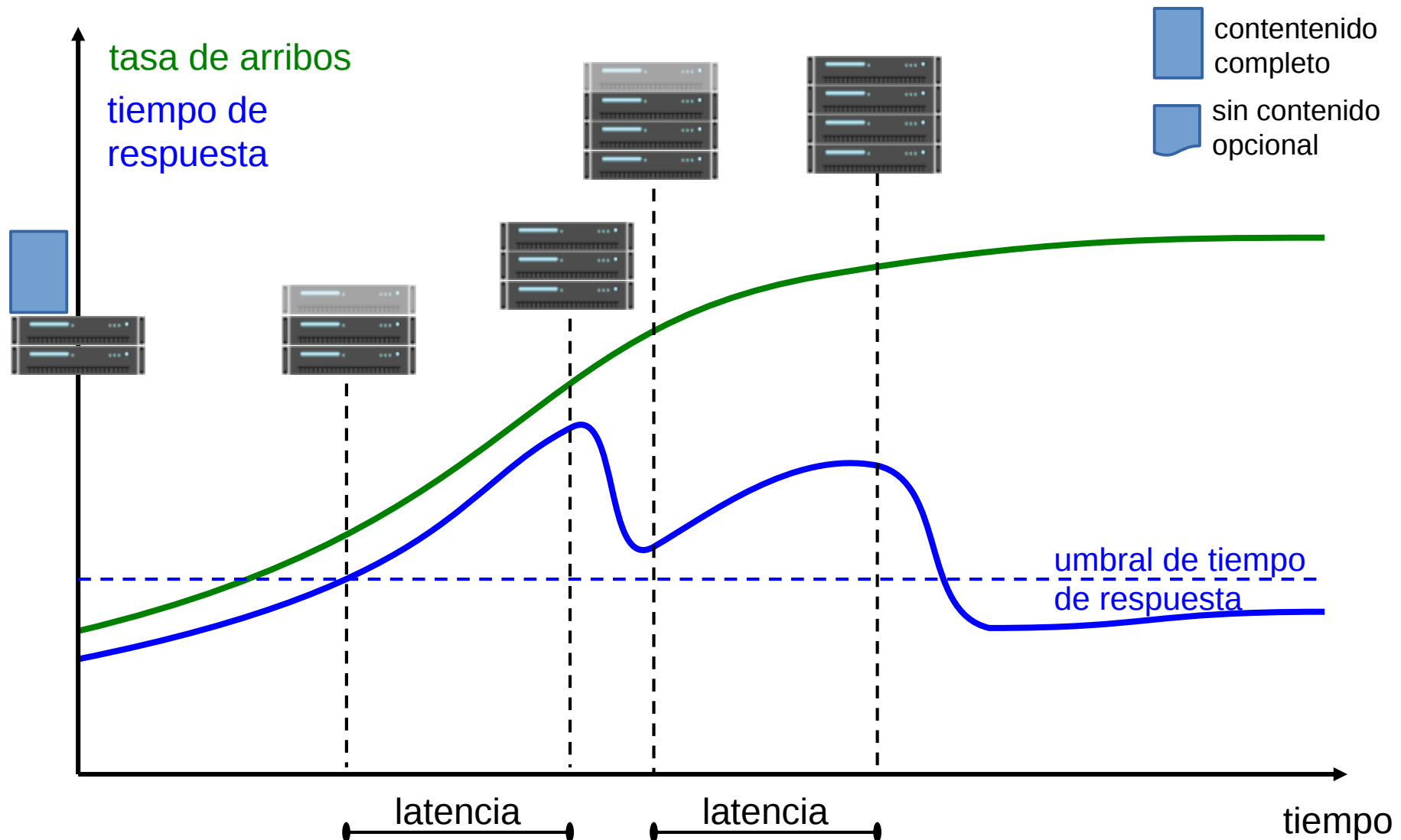
Decisión basada en utilidad

Utilidad: una métrica que cuantifica la calidad de una alternativa según un criterio

$$U = \begin{cases} \tau a(dR_O + (1 - d)R_M) & \text{if } r \leq T \quad \longleftarrow \text{ premio} \\ \tau \min(0, a - \kappa)R_O & \text{if } r > T \quad \longleftarrow \text{ penalidad} \end{cases}$$

Algoritmo: filtrar alternativas no-aplicables, cuantificar la utilidad de cada alternativa, elegir la de mayor utilidad

Ejemplo de Adaptación Reactiva



Adaptación PLA (Proactive Latency-Aware)

Mejora la efectividad de la auto-adaptación

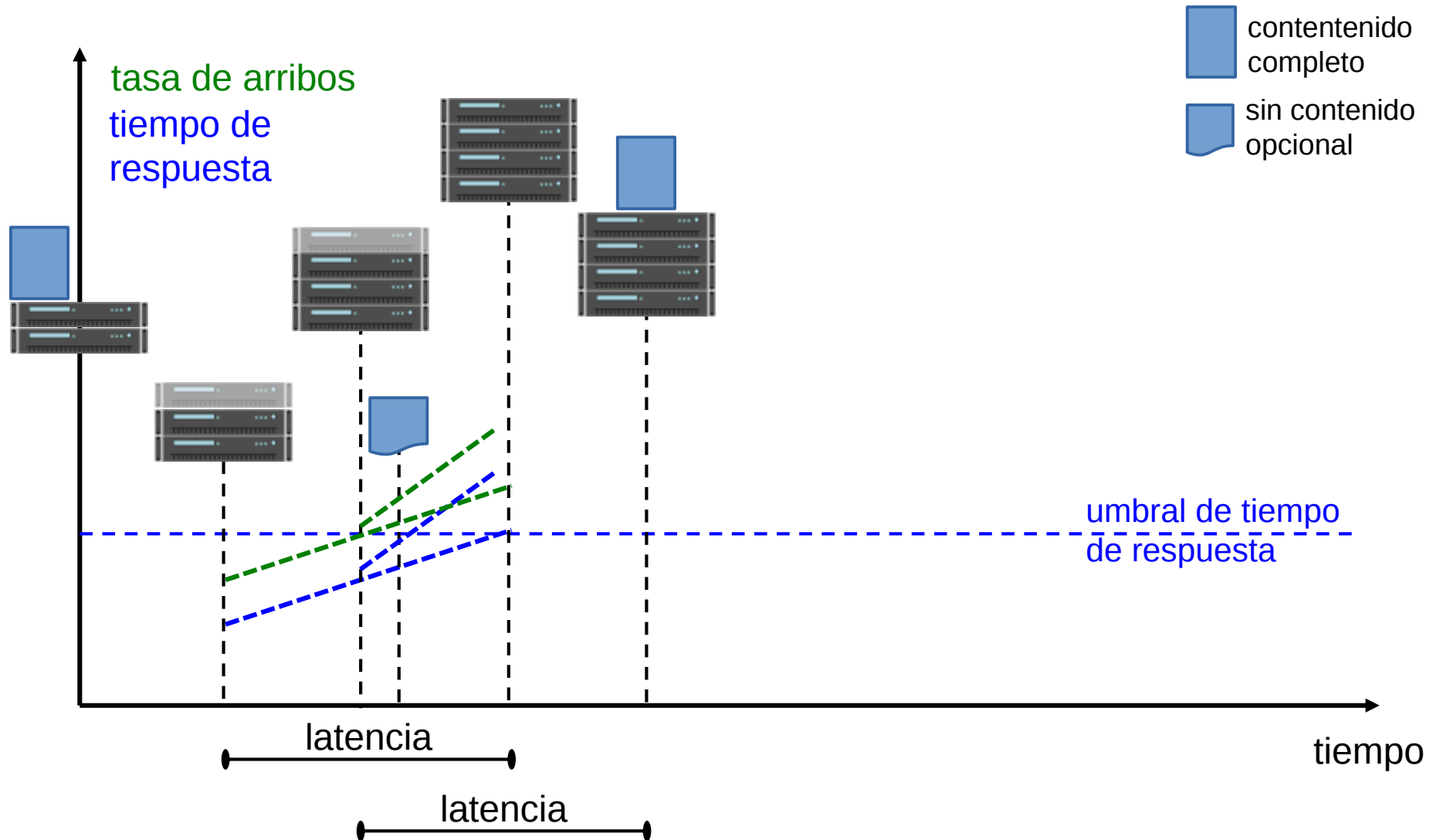
- usando predicciones del entorno para adaptarse proactivamente
- teniendo en cuenta la latencia de las tácticas cuando toma la decisión

La decisión de adaptación es un proceso de decisiones secuenciales:

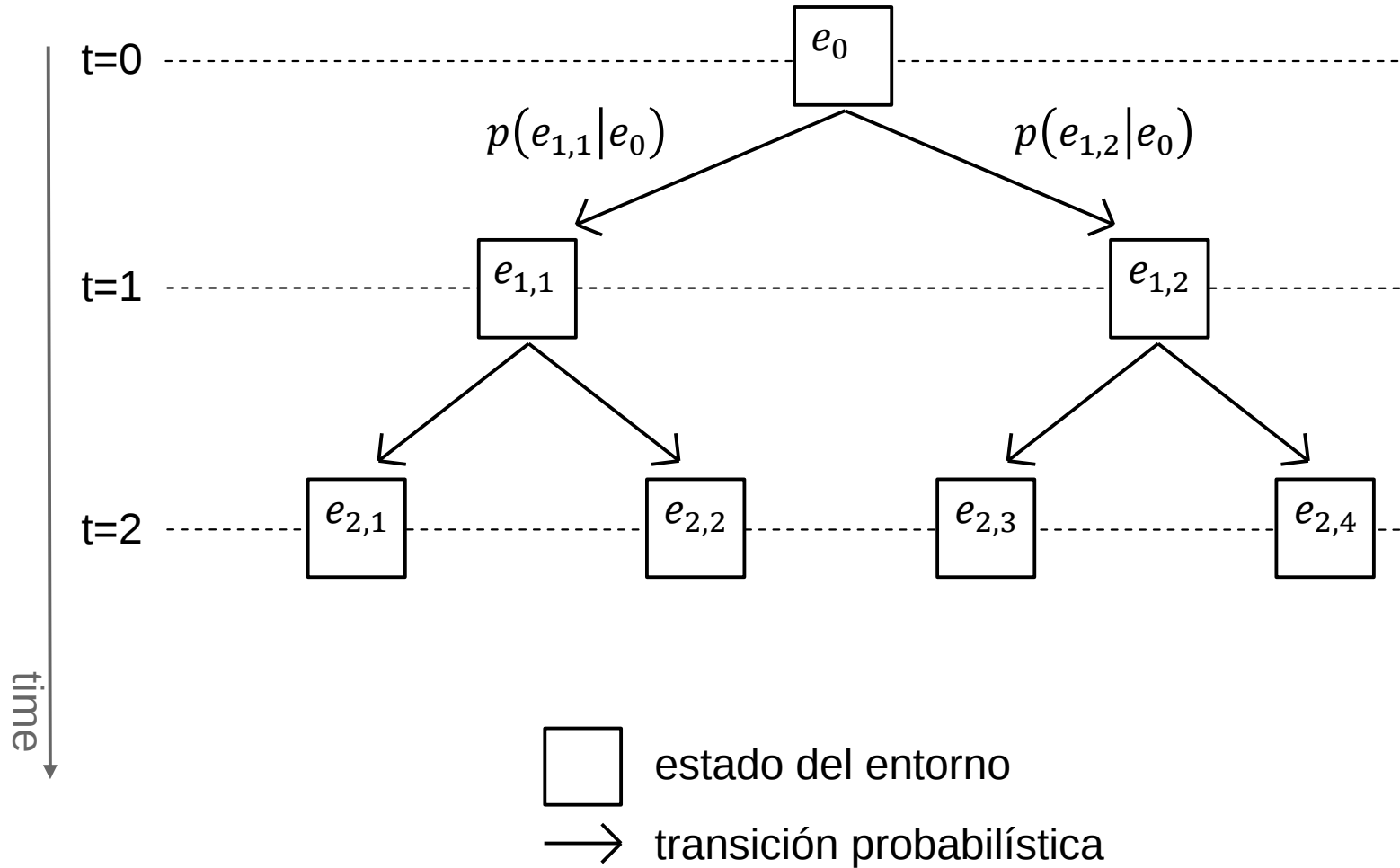
- selecciona las acciones de adaptación para maximizar la utilidad esperada en el contexto de un entorno con incierto

G. Moreno, J. Camara, D. Garlan and B. Schmerl. "Proactive Self-Adaptation under Uncertainty: A Probabilistic Model Checking Approach." In *proceedings of the Joint Meeting of the European Software Engineering Conference and the Symposium on Foundations of Software Engineering*, 2015

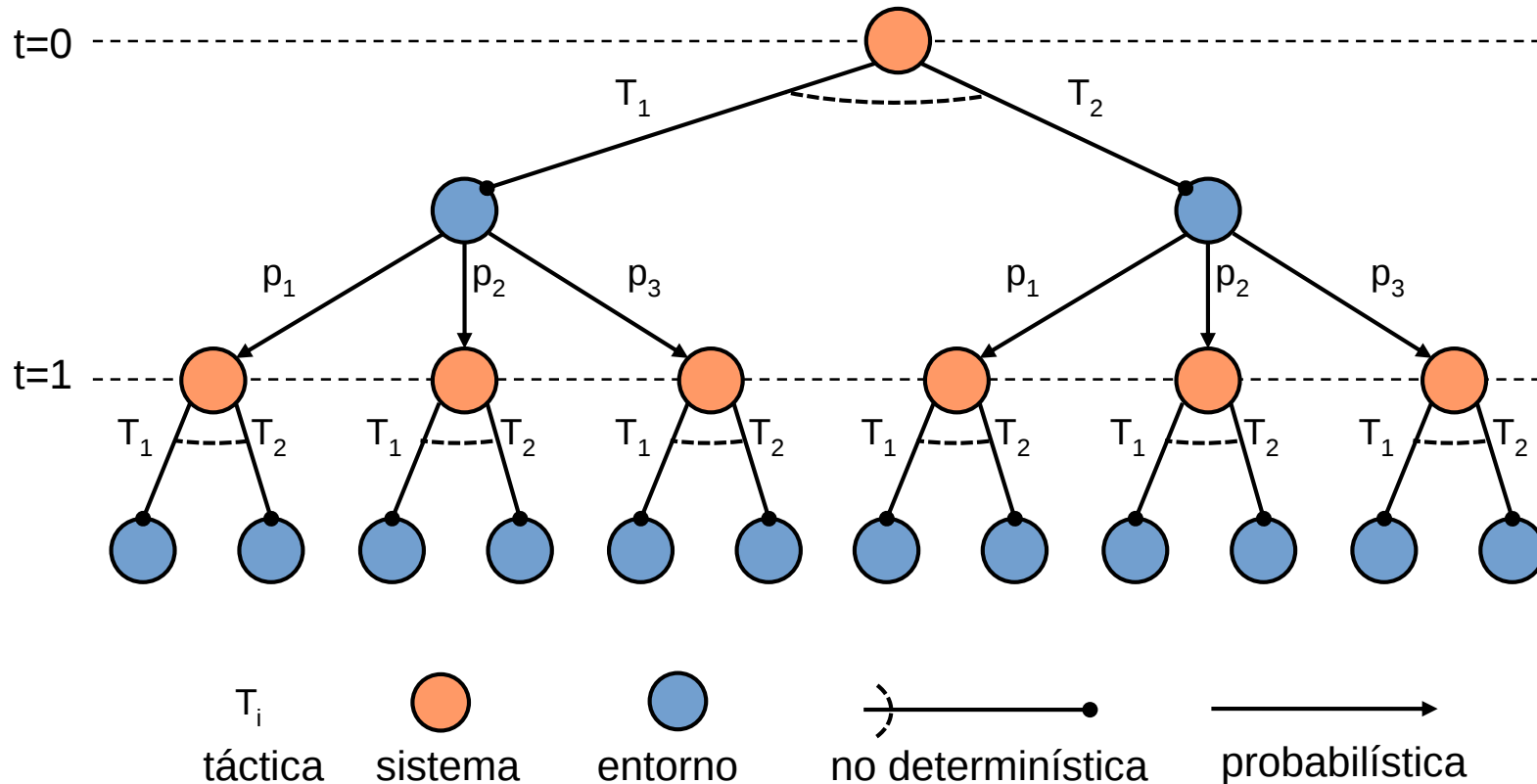
Adaptación Proactiva



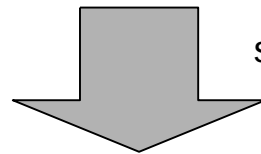
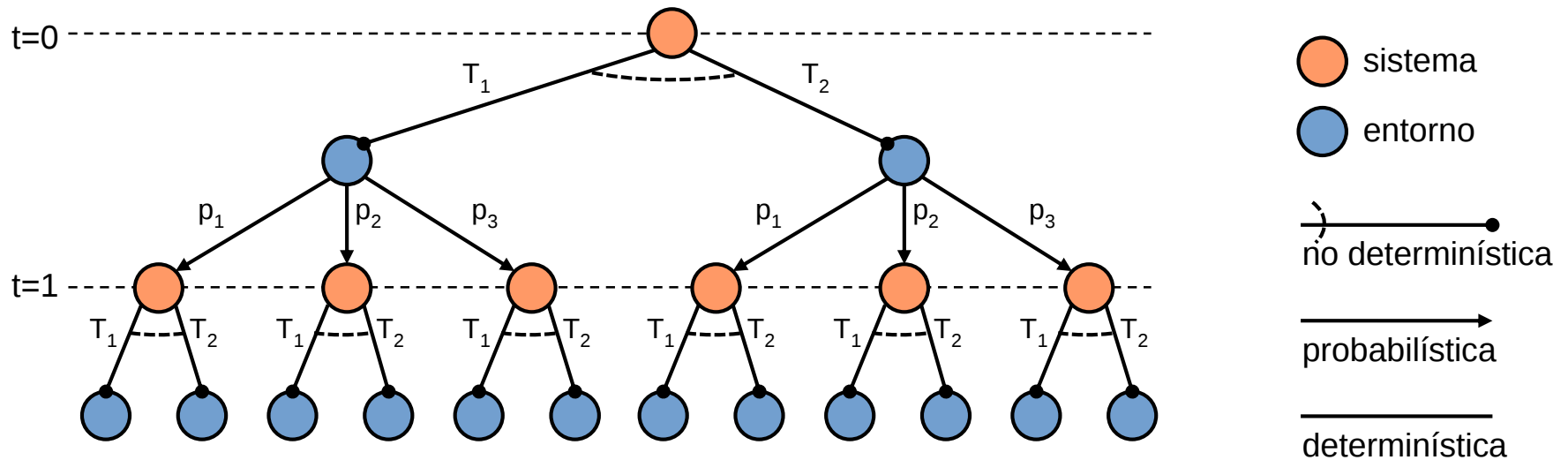
Modelo Estocástico del Entorno



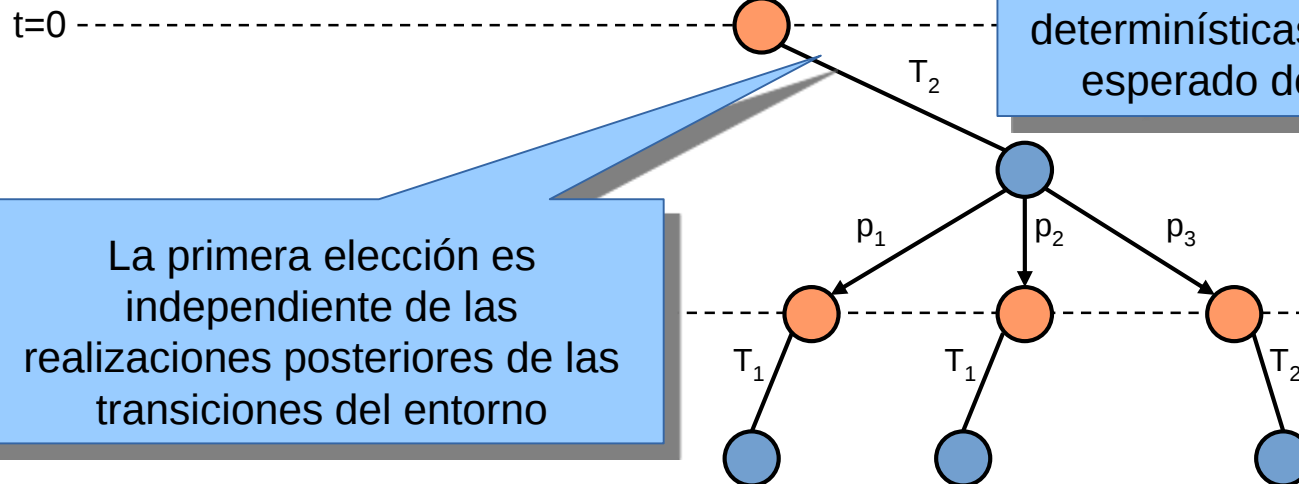
Decisión con Proceso de Decisión de Markov



Decisiones de adaptación modeladas como elecciones no determinísticas



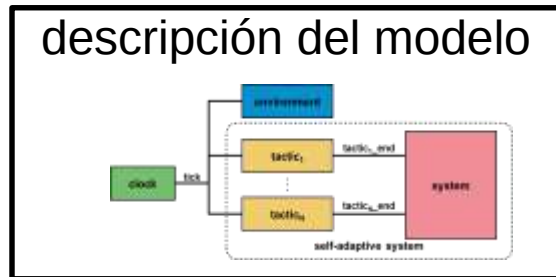
síntesis de política óptima



La primera elección es independiente de las realizaciones posteriores de las transiciones del entorno

Resuelve las elecciones no determinísticas para maximizar el valor esperado de la función de utilidad

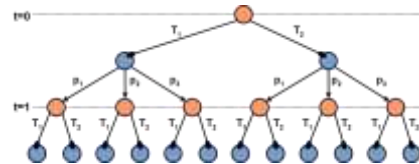
Verificación de Modelos Probabilísticos



propiedad en lógica temporal probabilística

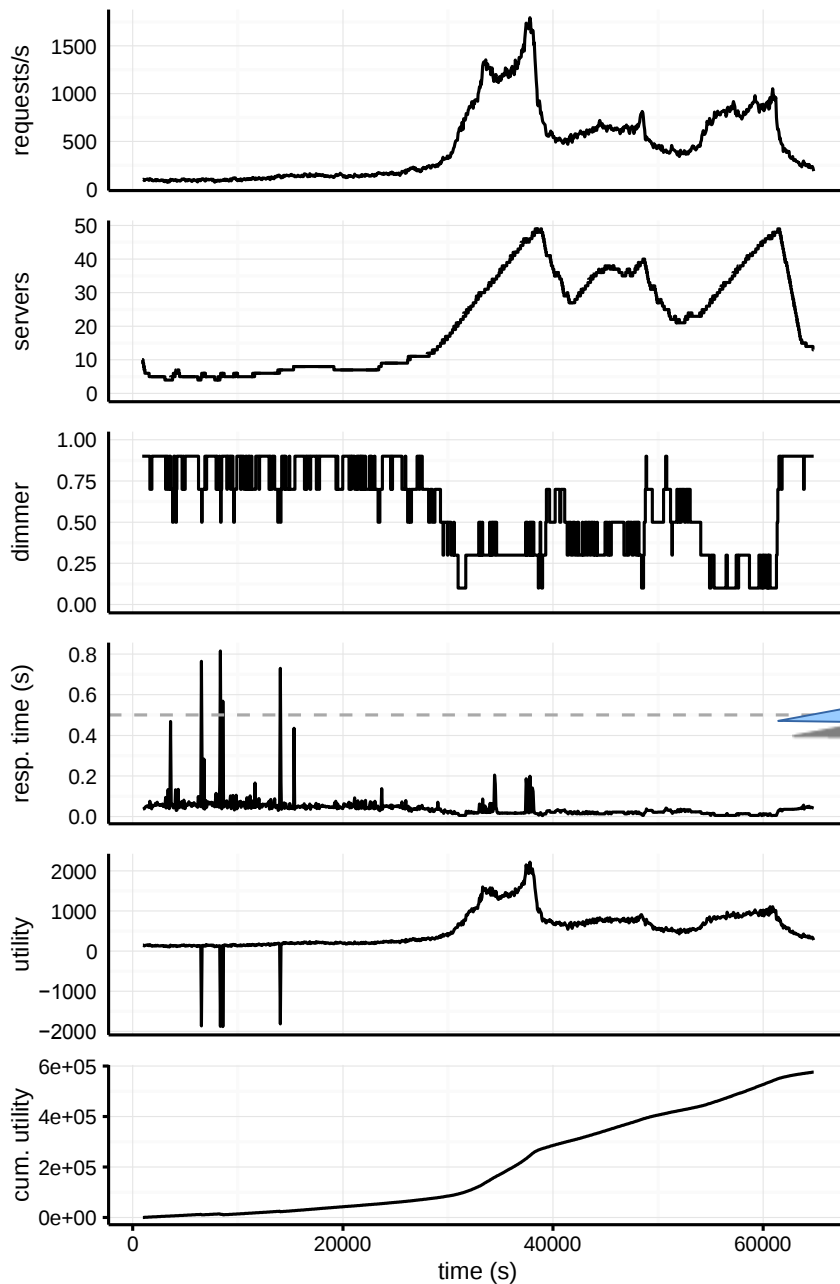
$$R_{max=?}^{util} [F \text{ horizon}]$$

PRISM model checker



optimal
policy

M. Kwiatkowska, G. Norman, and D. Parker. "PRISM 4.0: verification of probabilistic real-time systems." In International Conference on Computer Aided Verification, 2011.



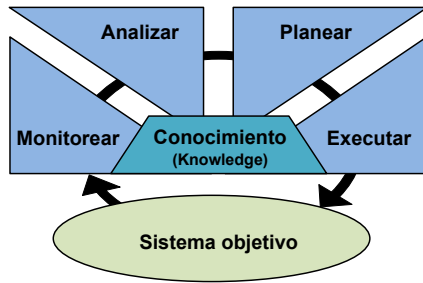
Datos: 18 horas de tráfico a un cluster regional del Mundial '98

Tamaño del cluster: 60 servidores

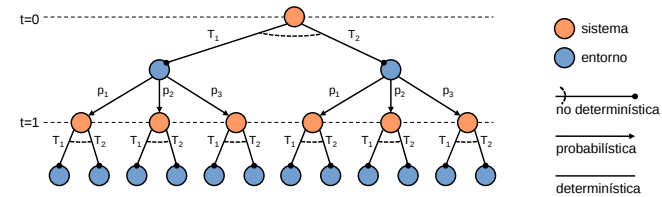
Decisiones: 1 cada 60 seg.

Solo 4 violaciones
del tiempo de
respuesta
Tiempo promedio
de decisión 8 seg.

MAPE-K



J.O. Kephart, and D.M. Chess. "The vision of autonomic computing." *Computer* 36, no. 1, 2003



síntesis de política óptima

