

A Performance Comparison of Qos-Driven Service Selection Approaches

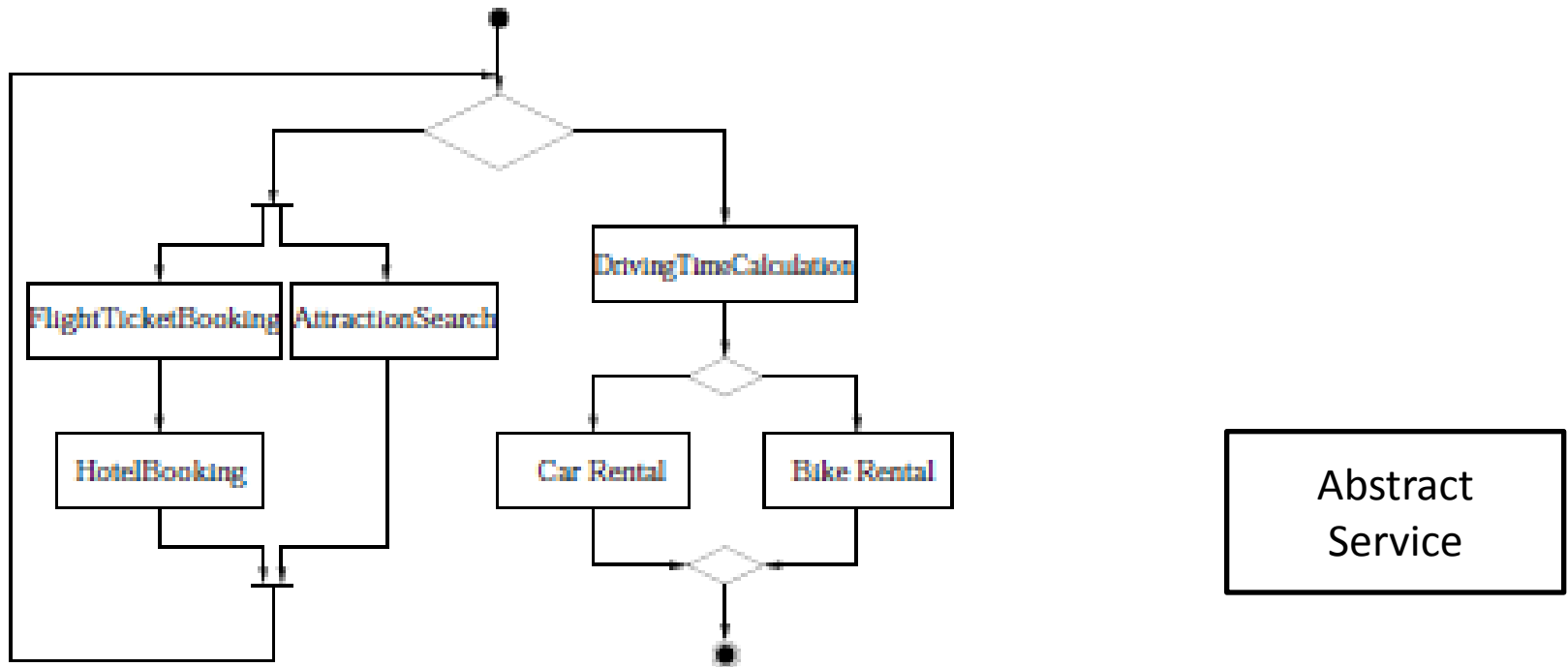
V. Cardellini, V. Di Valerio, V. Grassi, S.
Iannucci, F. Lo Presti

Talk Outline

- Background
 - SOA- composite service orchestrated by a broker
- Broker Service Selection Strategy
 - Per Request Approaches
 - Per Flow Approaches
- MOSES Implementation - Experimental Results
- Conclusions

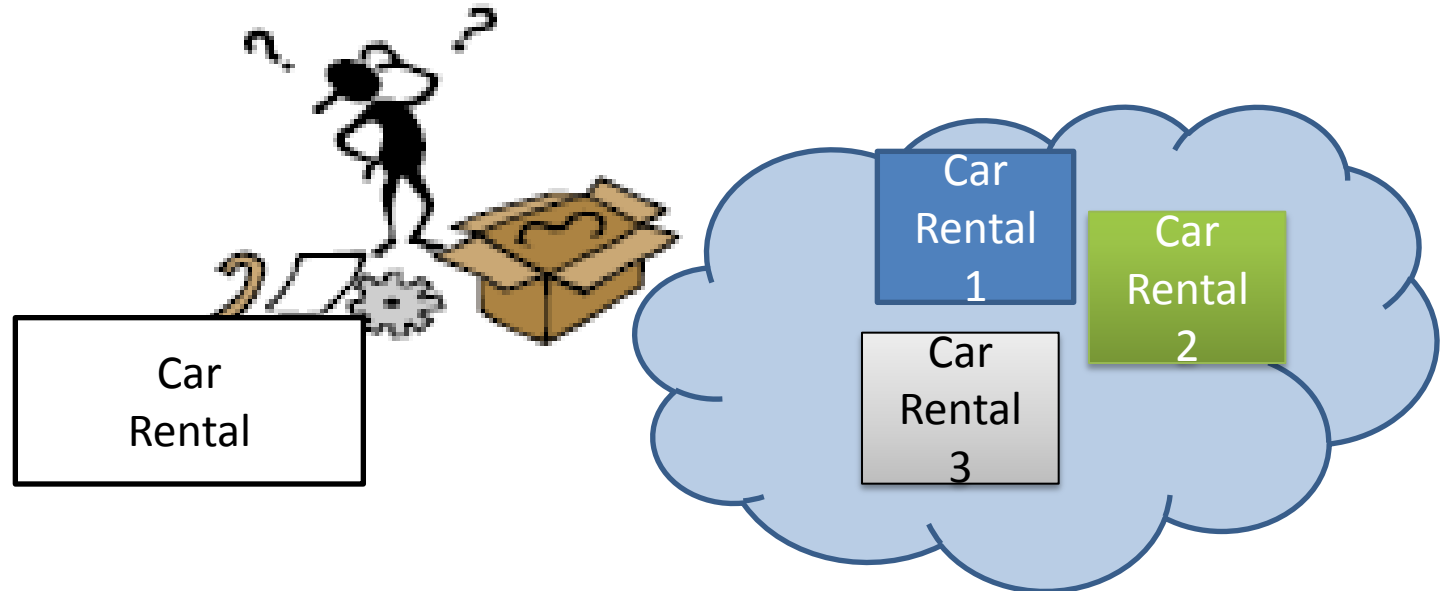
Motivation - SOA

- Composite service applications built using loosely coupled services
 - new added-value services built on top of existing third-party services, offering different QoS characteristics
- e.g, **Travel Planner service (workflow)**



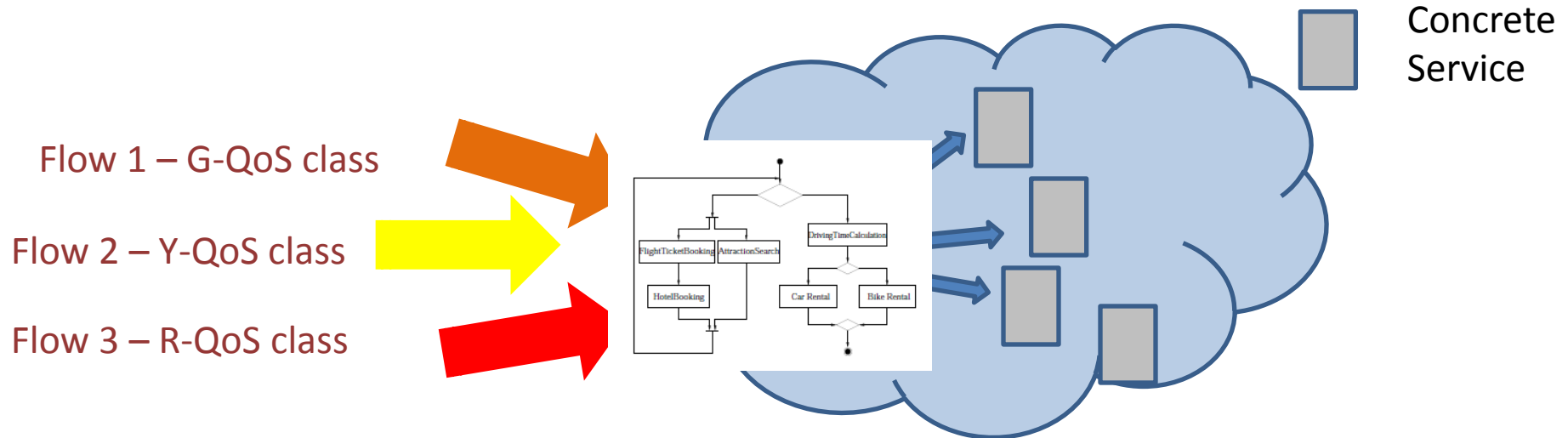
Motivation - SOA

- Composite service applications built using loosely coupled services
 - new added-value services built on top of existing third-party services, offering different QoS characteristics
- e.g, **Travel Planner service (workflow)**



Broker

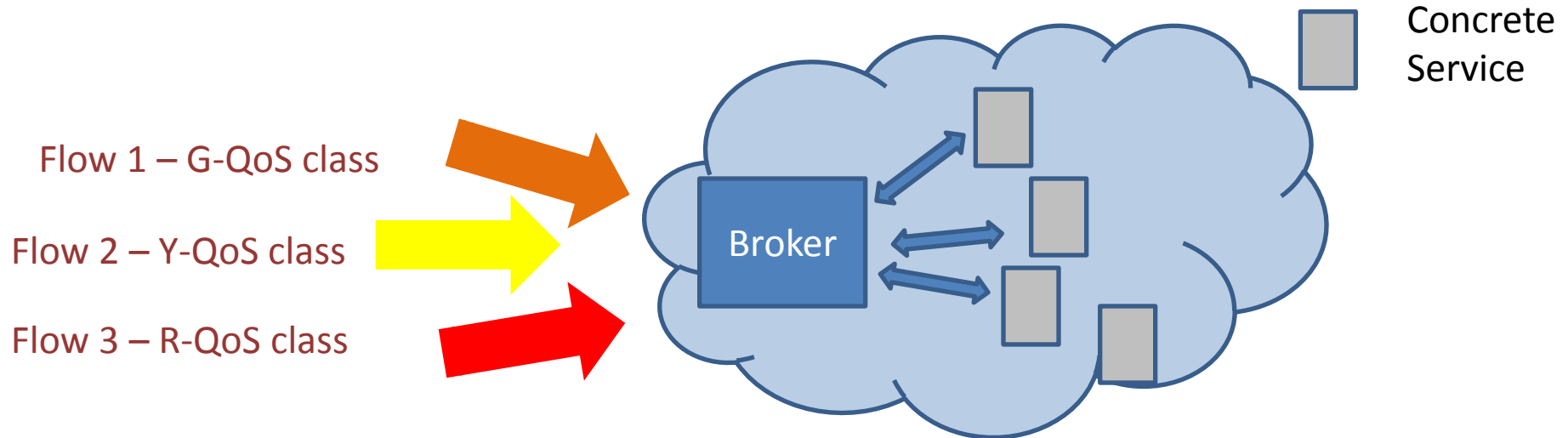
- We consider a broker that offers a composite service with multiple QoS classes to users generating requests



- **Dynamic** service *discovery*, *selection/adaptation* and *composition*
 - “service market”: several services offering the same functionality, e.g., hotel booking, with different QoS/cost

Broker

- We consider a broker that offers a composite service with multiple QoS classes to users generating requests

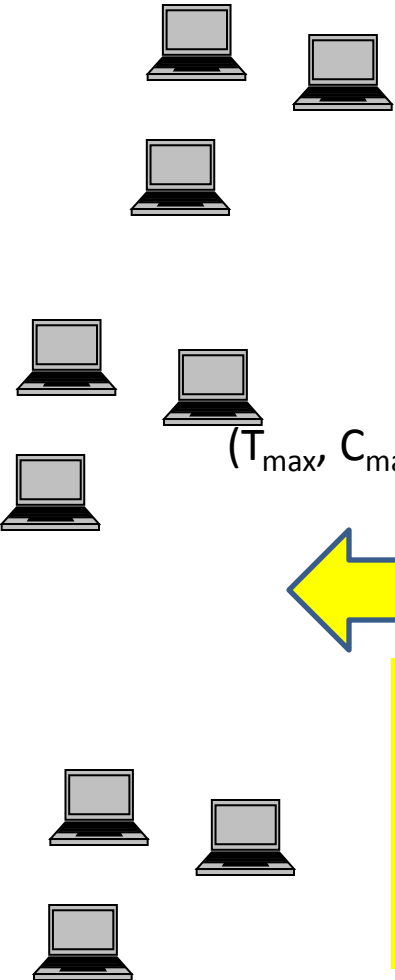


Service Selection Strategy: We want to **optimize** a suitable broker utility function while guaranteeing user QoS using available concrete services

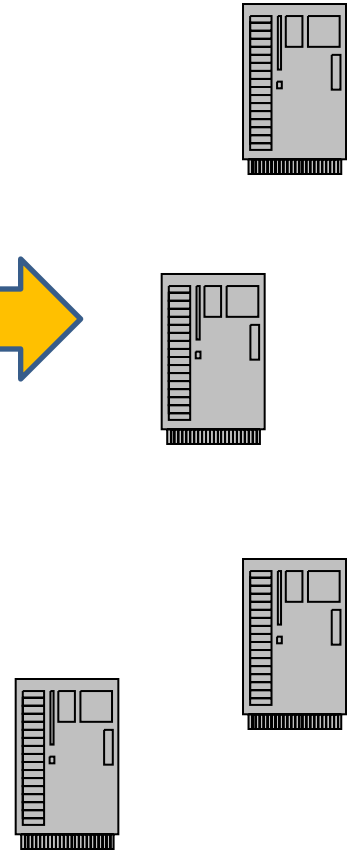
Broker: SLA

Negotiate SLA with service providers, establishing for each SP the value of QoS attributes (t,c,a)

Clients

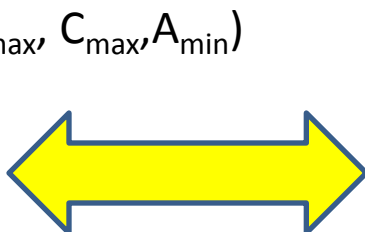


SPs

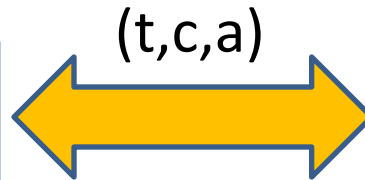


BROKER

• Acts as an intermediary between service requestors and providers



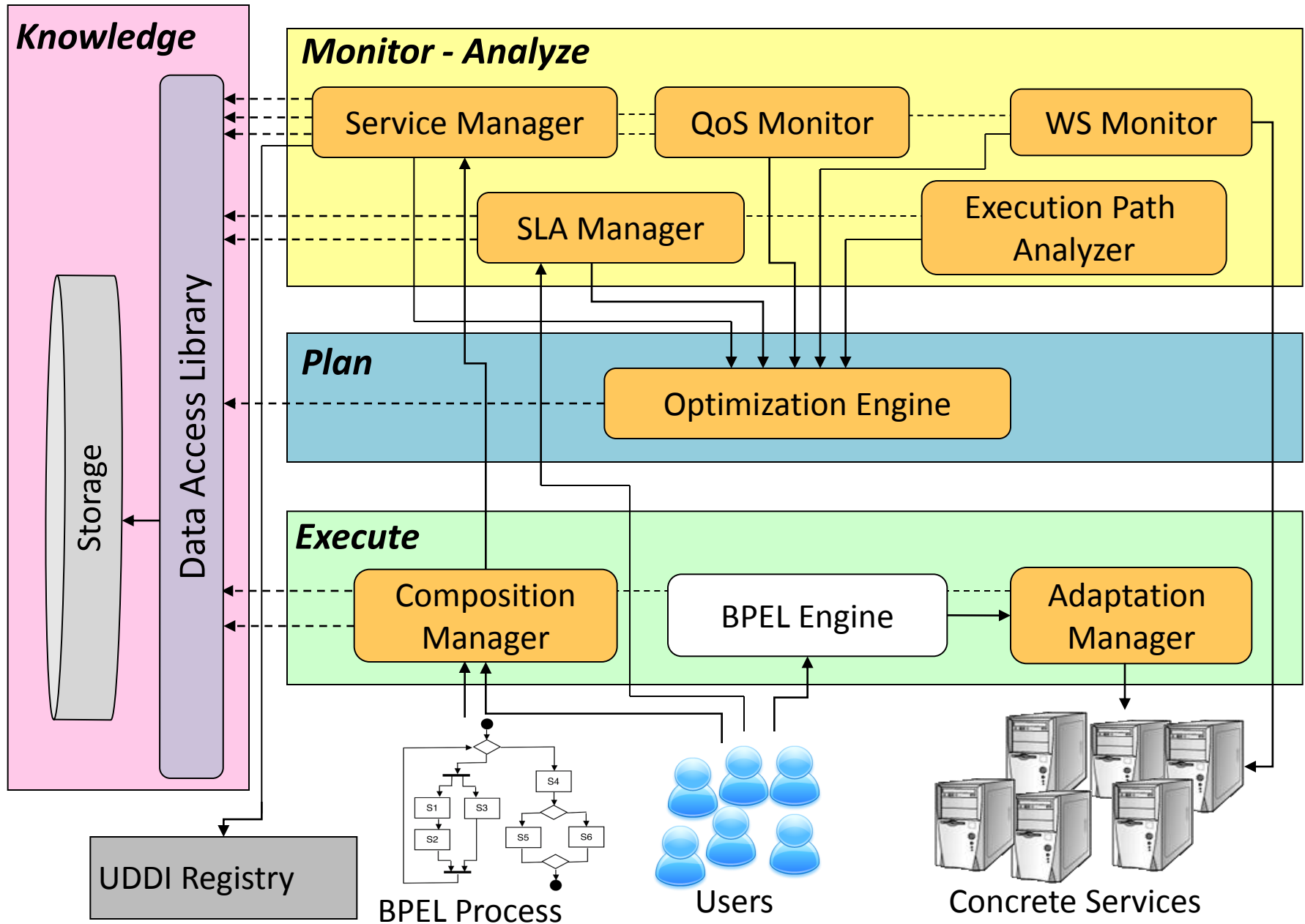
$(T_{max}, C_{max}, A_{min})$



(t,c,a)

Negotiate SLA with each requestor establishing the offered QoS level of the composite service $(T_{max}, C_{max}, A_{min})$

Broker architecture overview: Moses



Service Selection Strategies

- Per Request Approaches
 - Requests handled independently
 - Actual service determined on a per request basis
- Per Flow Approaches
 - Consider flows of requests generated by users/class
 - Probability of assigning a given service to a given request

Approaches Pro-Cons

- **Per-request**

- + Fine grain - per request Qos guarantee
- Computational Complexity - 0-1 problem
 - heuristics might be required
- Does not account for concrete service load
- How to handle high request rate?

- **Per-flow**

- Coarse grain - "on average" Qos guarantee
 - Notion of flow
 - a sequence of homogeneous requests originating by the same user organization, all requiring the same QoS level
- + Computational Complexity - LP formulation
- + Service Selection strategy need to be recomputed only when there are changes in the environment
 - Number of users, rate, concrete services, etc...

Per Request - Problem

Given

- A user request for the service with given QoS parameters
- Workflow with n abstract services: S_1, S_2, \dots, S_n
- for each S_i a set of concrete implementation s_{i1}, \dots, s_{im} , each characterized by its own QoS values

Determine for each S_i :

- the actual service s_{ij} which implements S_i as to satisfy user QoS and maximize a broker utility function

Per Request - Optimization Problem

- Find set $\{s_{ij}\}$ which maximizes the Broker Utility
 - weighted function of Qos metrics
 - e.g., minimize response time, maximize availability, a combination, etc.

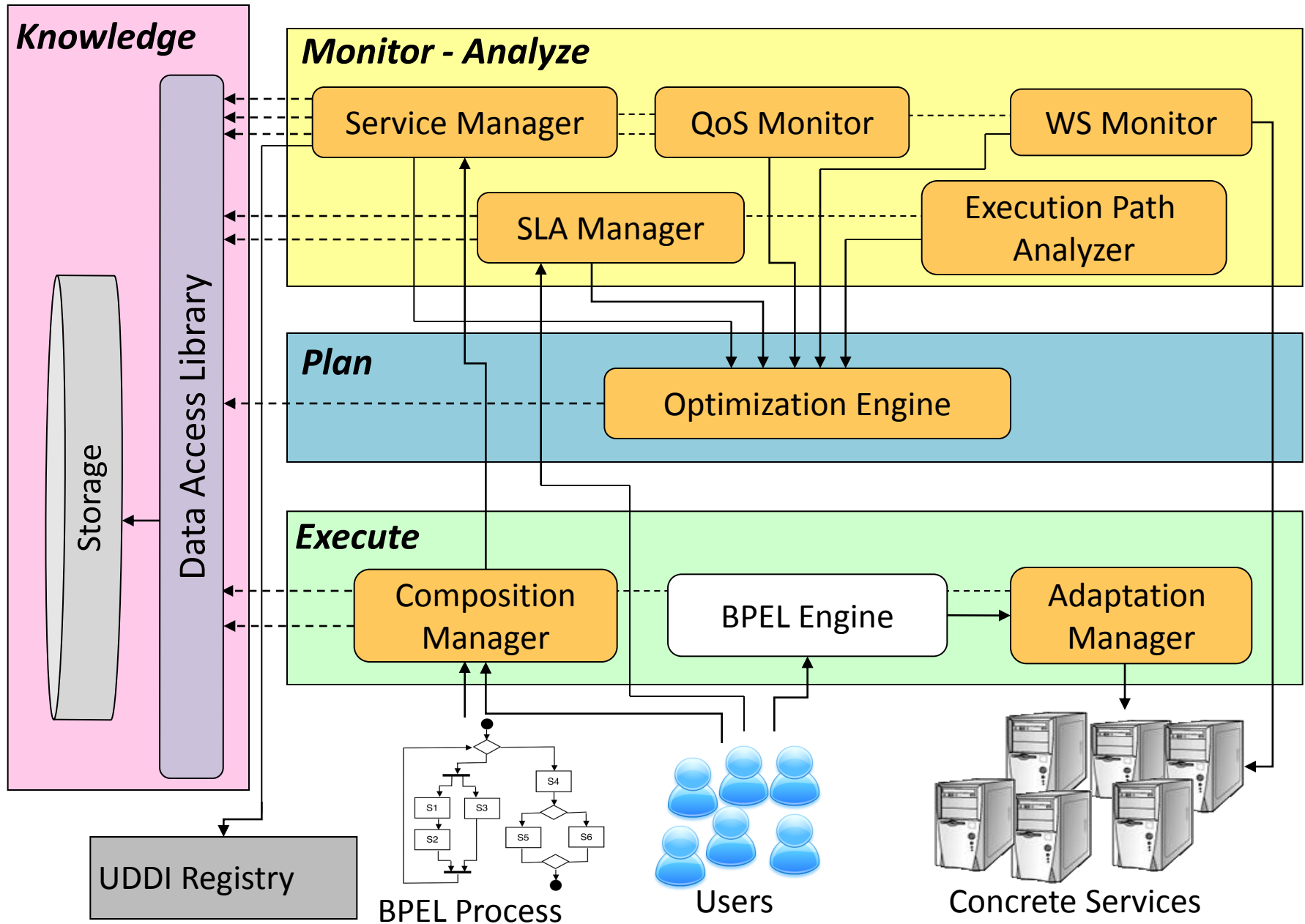
subject to:

$$\text{Response Time}(\{s_{ij}\}) \leq T_{\max}$$

$$\text{Cost}(\{s_{ij}\}) \leq C_{\max}$$

$$\text{Availability}(\{s_{ij}\}) \geq A_{\min}$$

Broker architecture overview: Moses



Per Flow - Problem

Given

- Set of classes of service with QoS parameters
- Workflow with n abstract services: S_1, S_2, \dots, S_n
- for each S_i a set of concrete implementation s_{i1}, \dots, s_{im} , each characterized by its own QoS values

Determine for each S_i and class of service k :

- Probabilities x_{ij}^k of assigning s_{ij} to a class k request which satisfy user QoS and maximizes a broker utility function

Per Flow - Optimization Problem

- Find $x = \{x^k\}_{k \text{ is a class}}$ which maximizes the Broker Utility
 - weighted function of Qos metrics(x)
 - e.g., minimize response time, maximize availability, a combination, etc.

subject to:

$$E[\text{Class } k \text{ Response Time}(x)] \leq T_{\max}^k$$

$$E[\text{Class } k \text{ Cost}(x)] \leq C_{\max}^k$$

$$\text{Class } k \text{ availability}(x) \geq A_{\min}^k$$

$$\text{Concrete service } s_{ij} \text{ load}(x) \leq L_{ij}$$

$$0 \leq x \leq 1$$

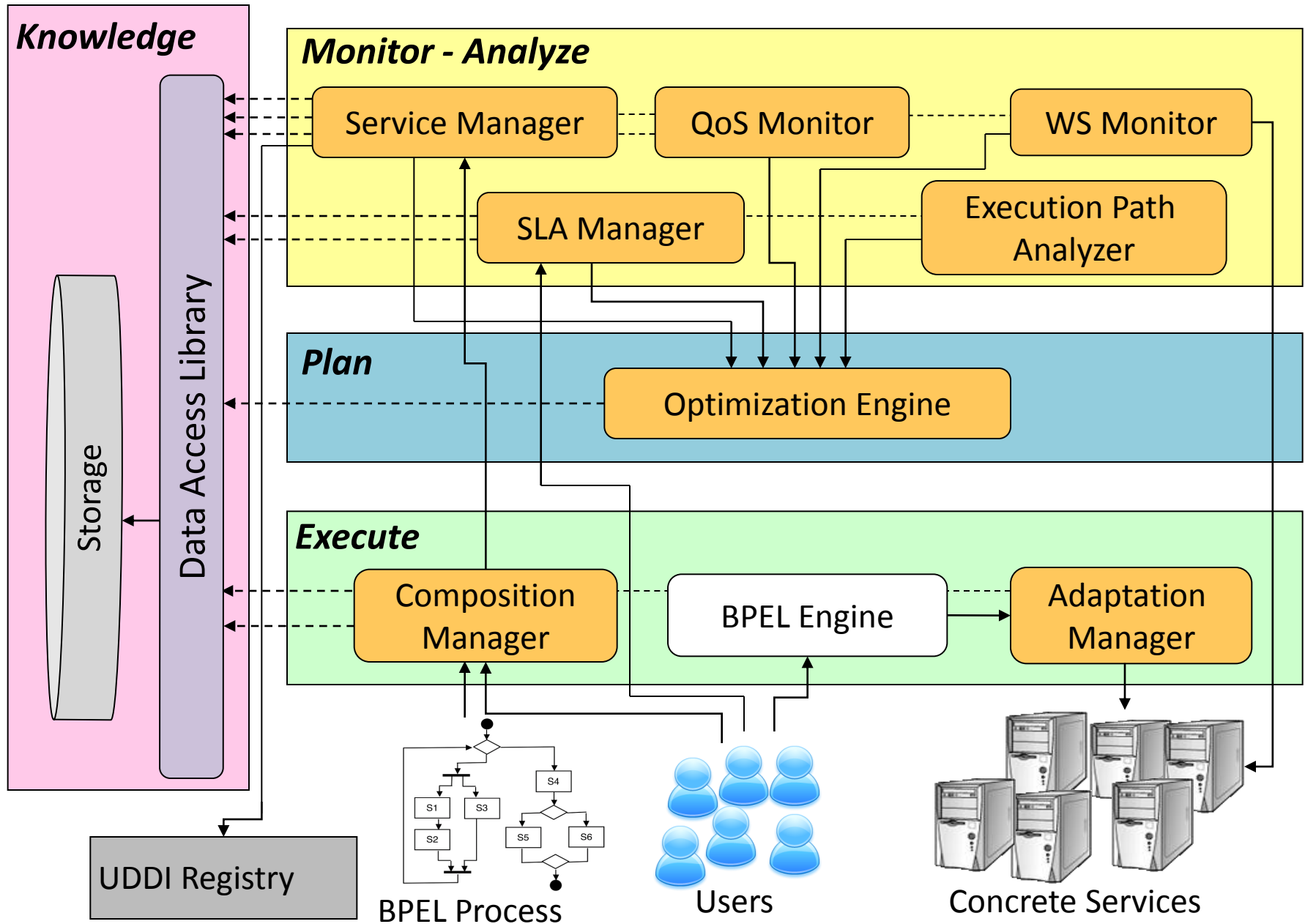
Per Flow -Service Selection Strategy

- Adaptation strategy for class k user is a vector $x^k = \{x^{k_1}, \dots, x^{k_n}\}$
- x^k_i strategy vector for service S_i
 - $x^k_i = \{x^{k_{is1}}, x^{k_{is2}}, \dots, x^{k_{ism}}\},$

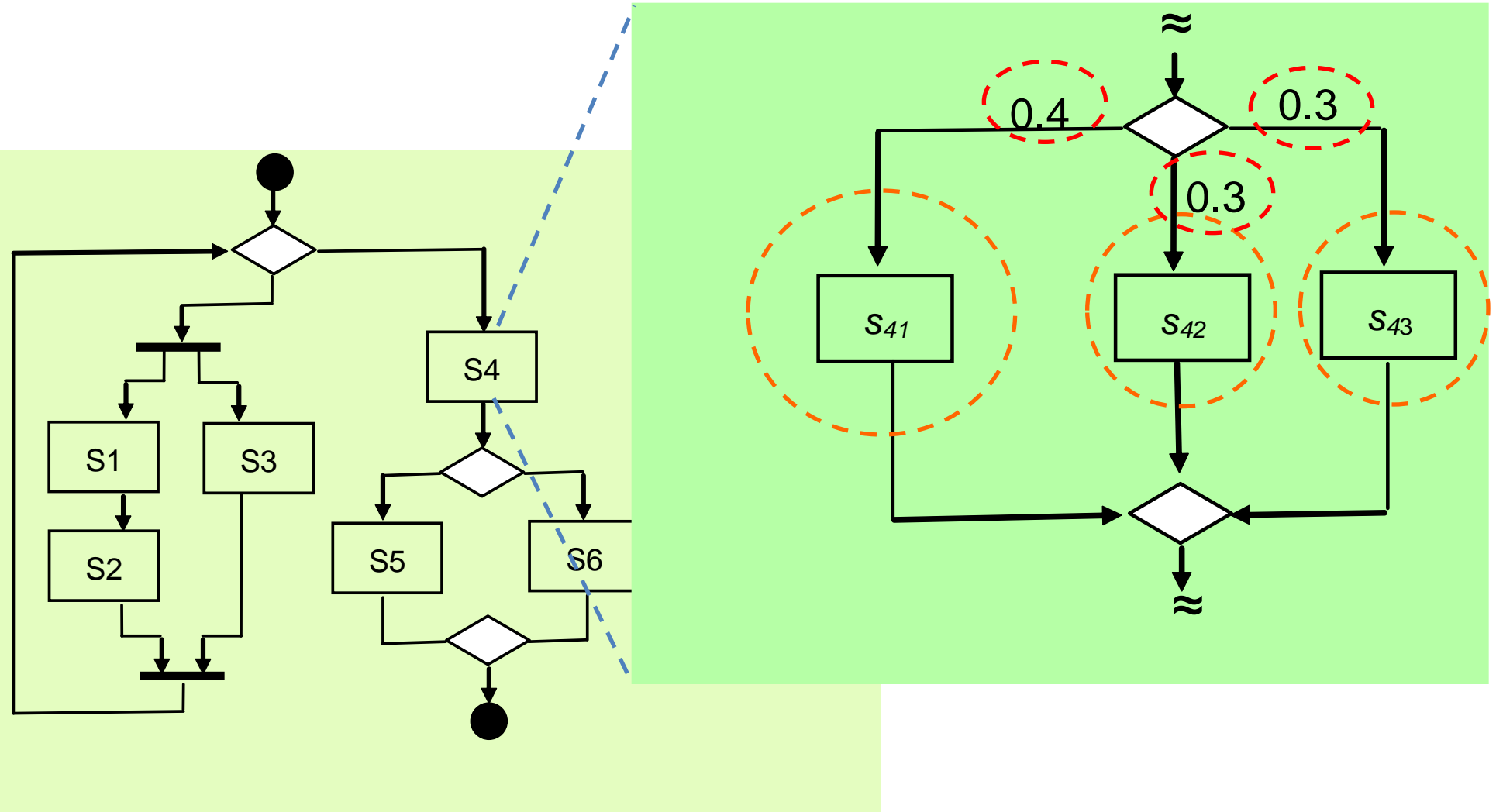
Strategy Implementation

- for each class k user request of service S_i , service s_{ij} is chosen with probability x^k_{ij}

Broker architecture overview: Moses



Per Flow - Service Selection Model



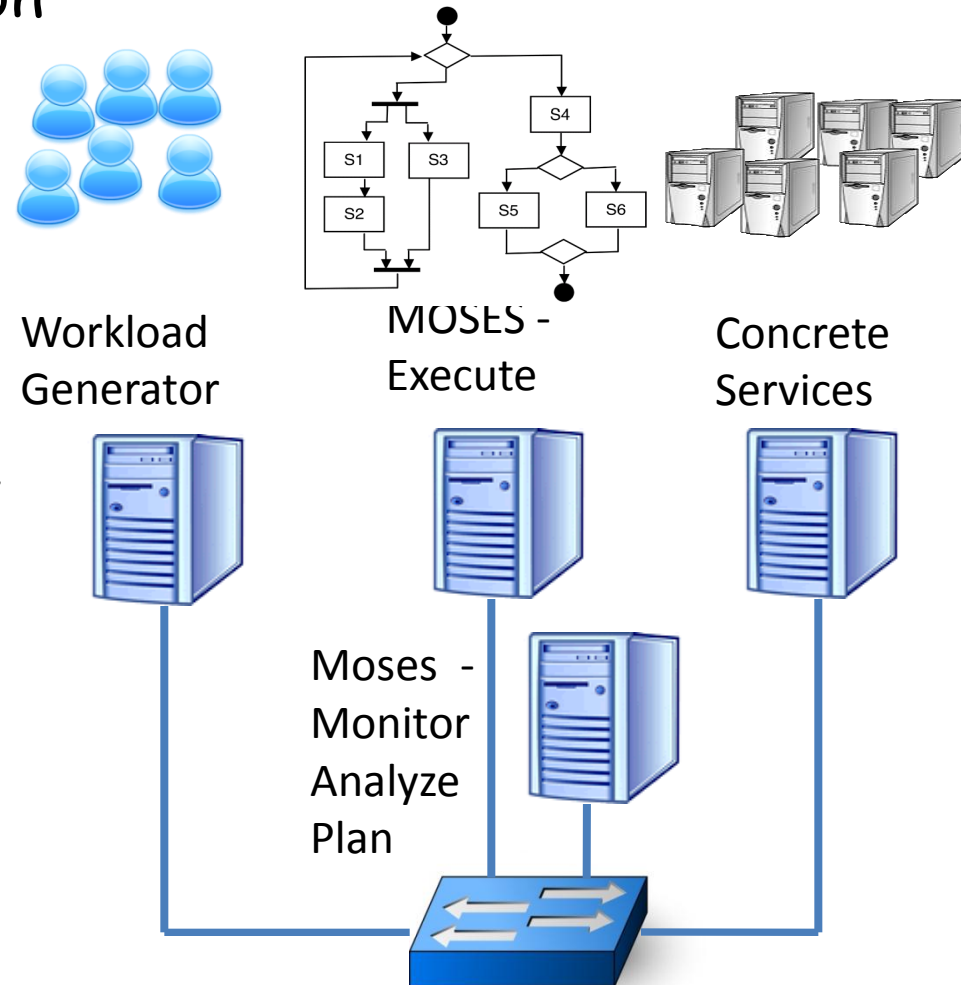
Experimental Results

- Implemented two Service Selection Approaches within MOSES

- Per Request [ArdagnaPernici07]
- Per Flow [Cardellini et al. 07]

- TravelPlanner Workflow

- 4-5 concrete service per abstract service
- Each service
 - handle up to 10 req/sec
 - Modelled as a M/G/1 queue
 - Has different Qos Parameters



Experimental Results

- Two sets of experiments

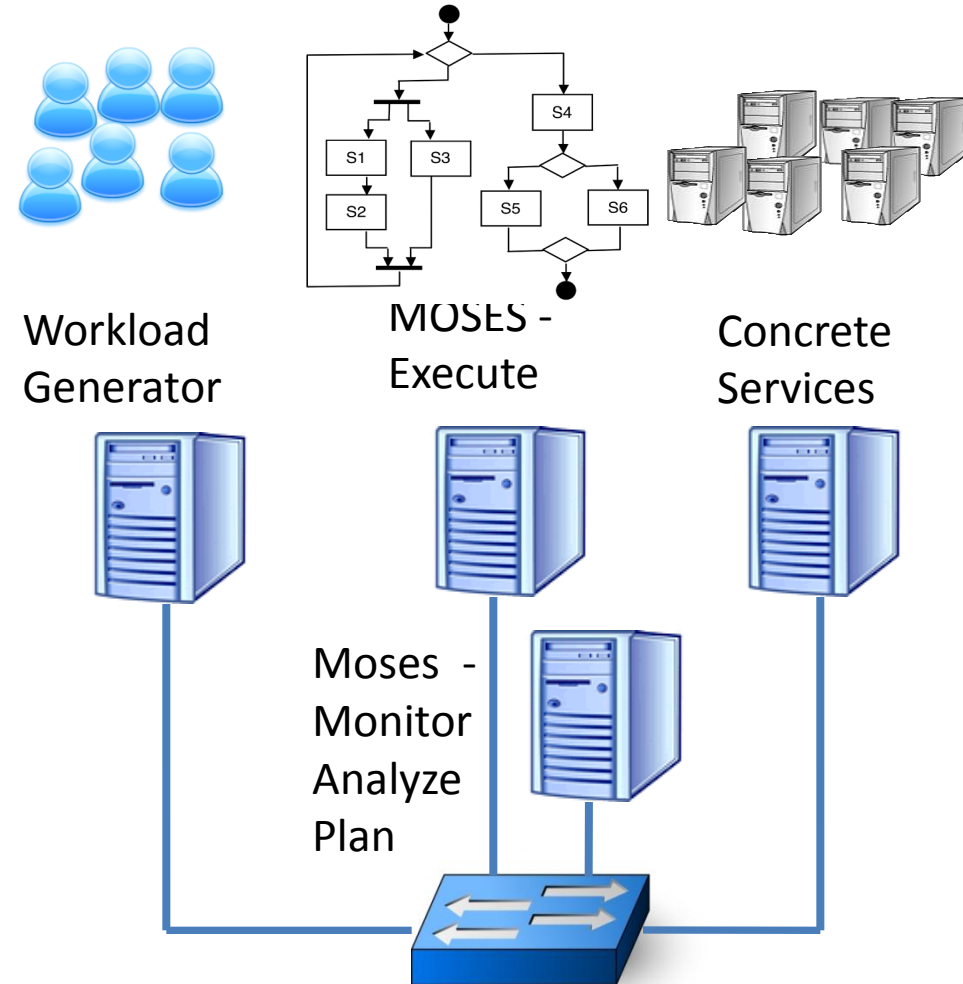
1. Stress Test

- Costant flow of request with same QoS requirements (one flow in the Per Flow approach)
- Increase request rate

2. Traffix mix scenario

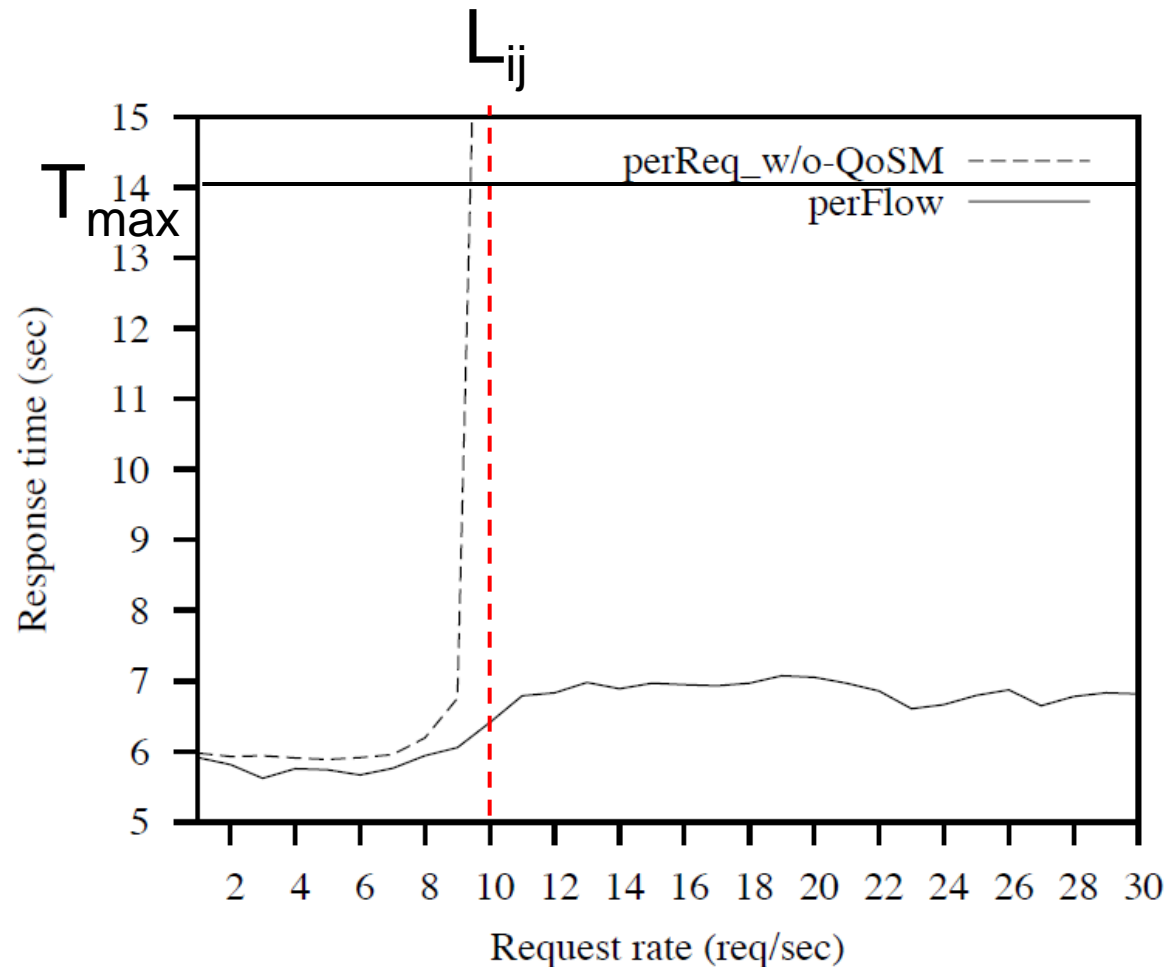
- 4 classes of users entering and leaving the system

- Broker Utility = Minimize Response Time



Ex. Results: Stress Test

- Response time vs. users request rate
- Same Qos for all requests
- Per request
 - provides always the same solution
 - works until the bottleneck service saturates
- Per flow
 - Split the load among the different available service
 - Scales better



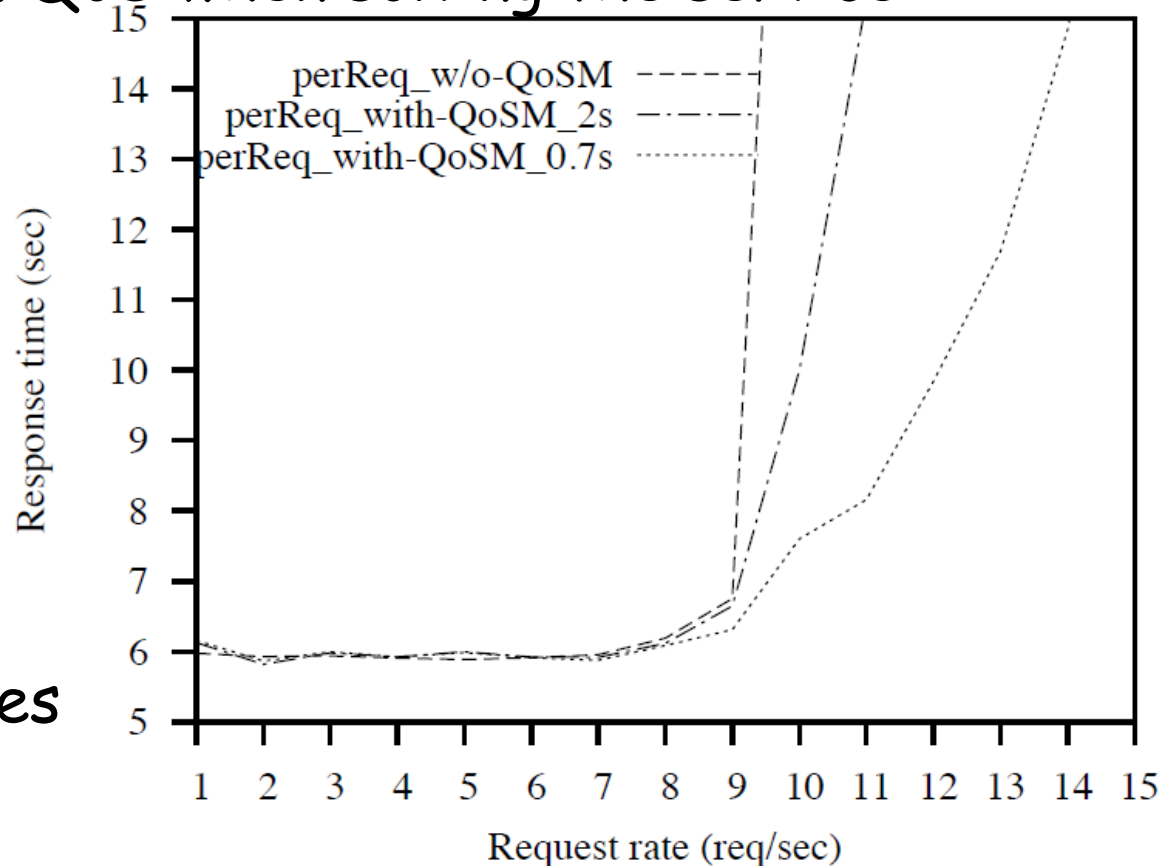
Ex. Results: Stress Test

- What if we use the (actual) monitored service QoS rather than the (SLA based) agreed on QoS when solving the service selection problem?

⇒ As a service gets close to saturation it not selected anymore

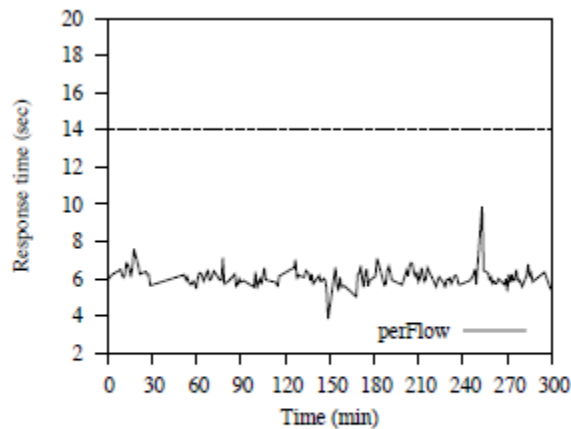
⇒ Per request improves

...but monitoring cost increases

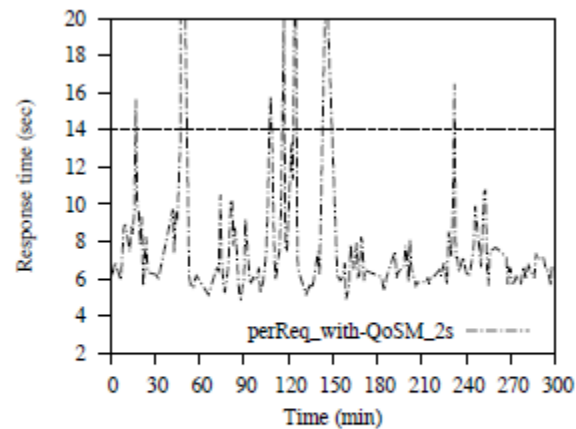


Ex Results: Traffic mix scenario

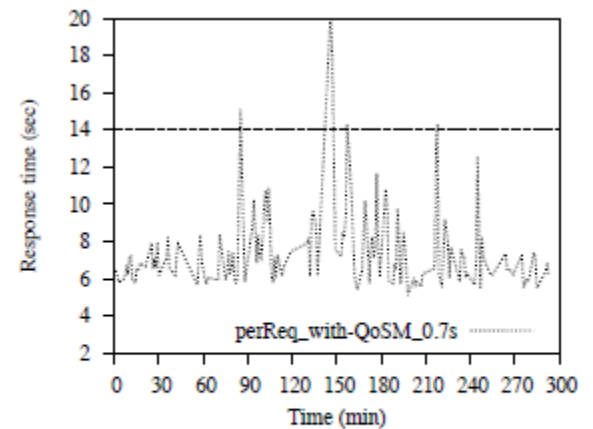
- Users arrive/leave the system generating flow of requests
 - 4 service classes with different QoS
 - 6.75 req/sec on average
- Results from the most stringent QoS class users



(a) Per-flow approach



(b) Per-request approach with QoS Monitor every 2s



(c) Per-request approach with QoS Monitor every 0.7s

Conclusions and Future Work

- Compared "Per Request" and "Per Flow" Approaches to service selection
- Assess the relative benefits
- Future
 - Per request "Load Aware" [SOSE11]
 - Call admission next talk ;-)
 - Multiple selfish Brokers scenario: game theory