A Framework for QoS-Aware Software Components

Daniel A. Menascé
menasce@cs.gmu.edu

Honglei Ruan
hruan@cs.gmu.edu

Hassan Gomaa
hgomaa@ise.gmu.edu

George Mason University, Fairfax, VA 22030

© 2004 D. A. Menascé. All Rights Reserved.

Motivation

- Characteristics of new generation of complex software systems:
  - Highly distributed
  - Component-based
  - Service-oriented
  - Unsupervised operation
  - Hostile environments
  - Composed of a large number of "replaceable" components discovered at run-time
  - Run on a multitude of (unknown and heterogeneous) hardware and network platforms

© 2004 D. A. Menascé. All Rights Reserved.

Motivation (cont'd)

- Enabling technologies
  - Web Services:
    - SOAP, UDDI, WSDL
  - Grid Computing
  - Peer to Peer Networks
  - Wireless Networking

© 2004 D. A. Menascé. All Rights Reserved.

Requirements of Next Generation Software Systems

- Adaptable and self-configurable to changes in workload intensity:
  - QoS requirements at the application and component level must be met.
- Adaptable and self-configurable to withstand attacks and failures:
  - Availability and security requirements must be met.

© 2004 D. A. Menascé. All Rights Reserved.

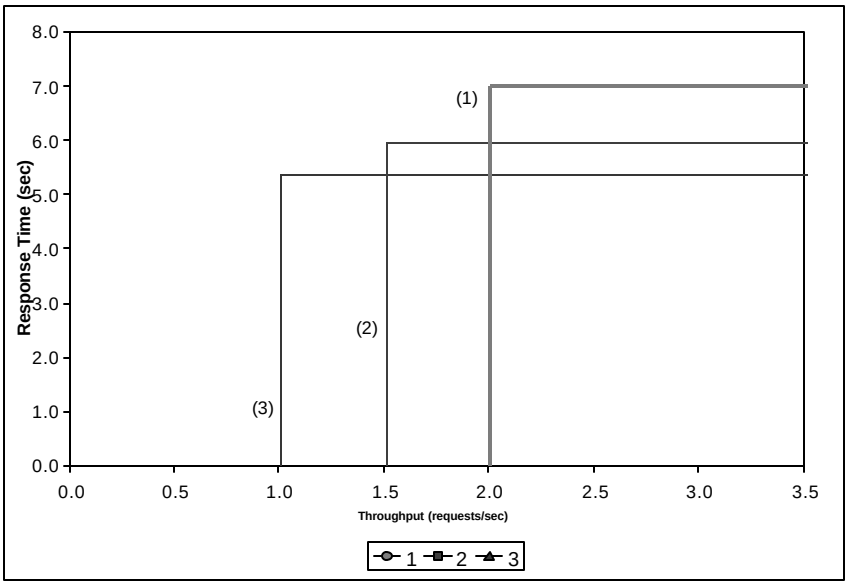
Requirements of Next Generation Software Systems

- Adaptable and self-configurable to changes in workload intensity:
  - QoS requirements at the application and component level must be met.
- Adaptable and self-configurable to withstand attacks and failures:
  - Availability and security requirements must be met.

5

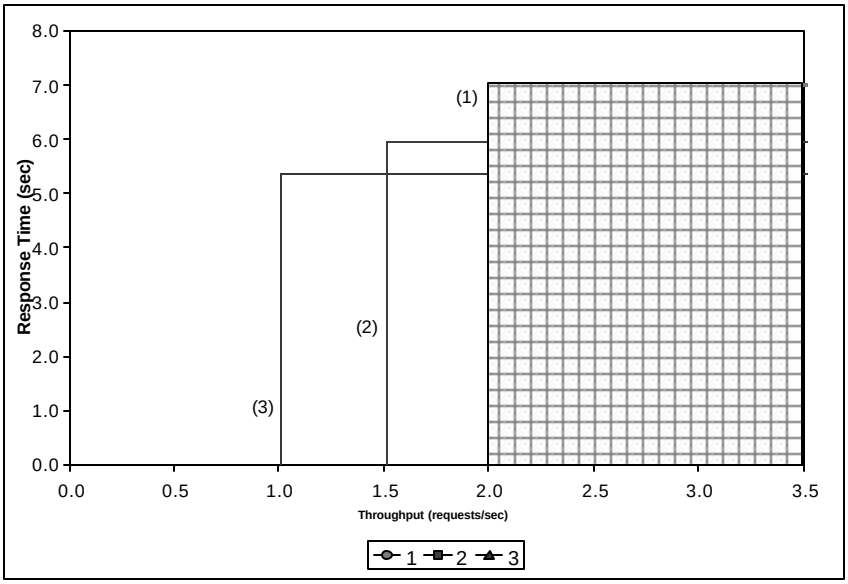
© 2004 D. A. Menascé. All Rights Reserved.

Feasibility Regions for a Q-component



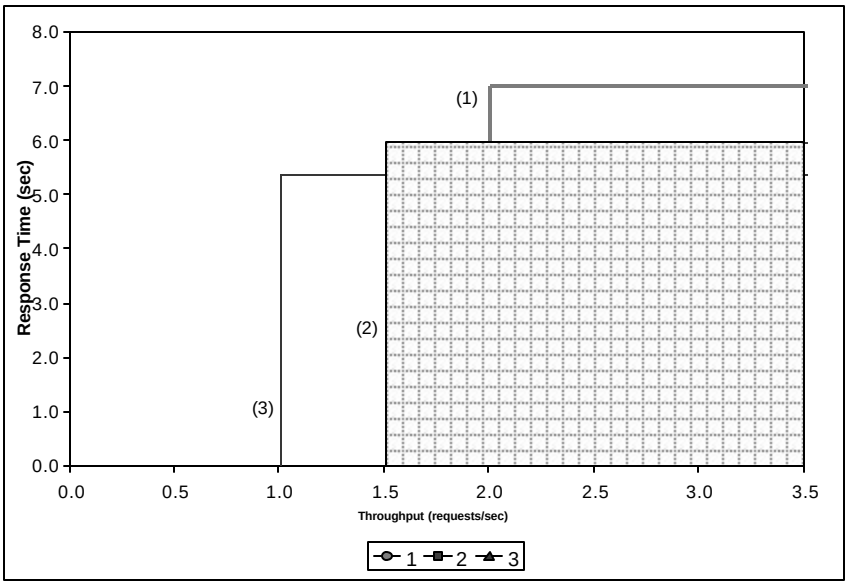
© 2004 D. A. Menascé. All Rights Reserved.

Feasibility Regions for a Q-component



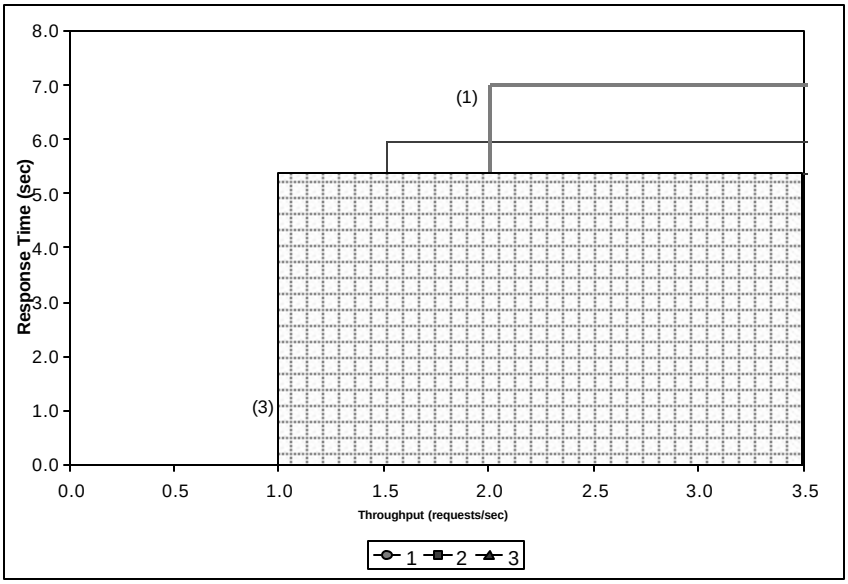
© 2004 D. A. Menascé. All Rights Reserved.

Feasibility Regions for a Q-component



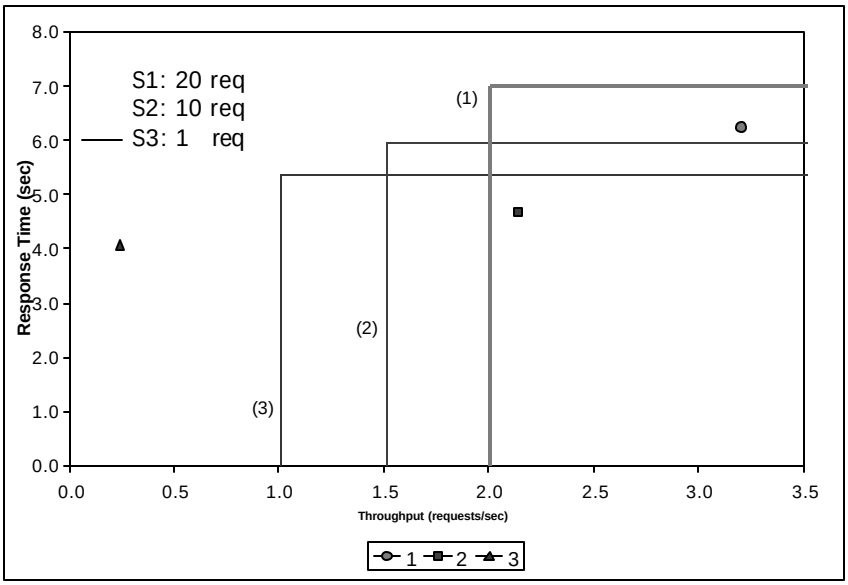
© 2004 D. A. Menascé. All Rights Reserved.

Feasibility Regions for a Q-component



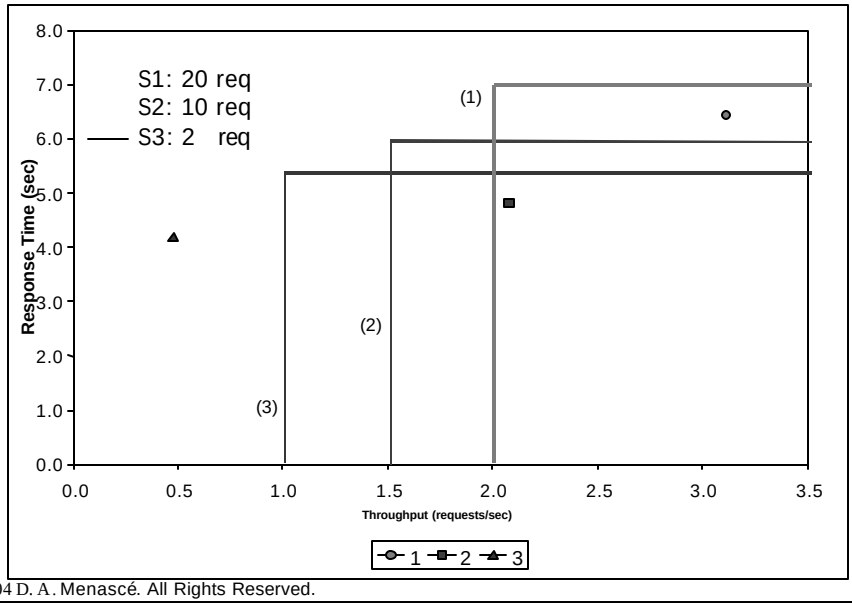
© 2004 D. A. Menascé. All Rights Reserved

Scenario 1: not feasible



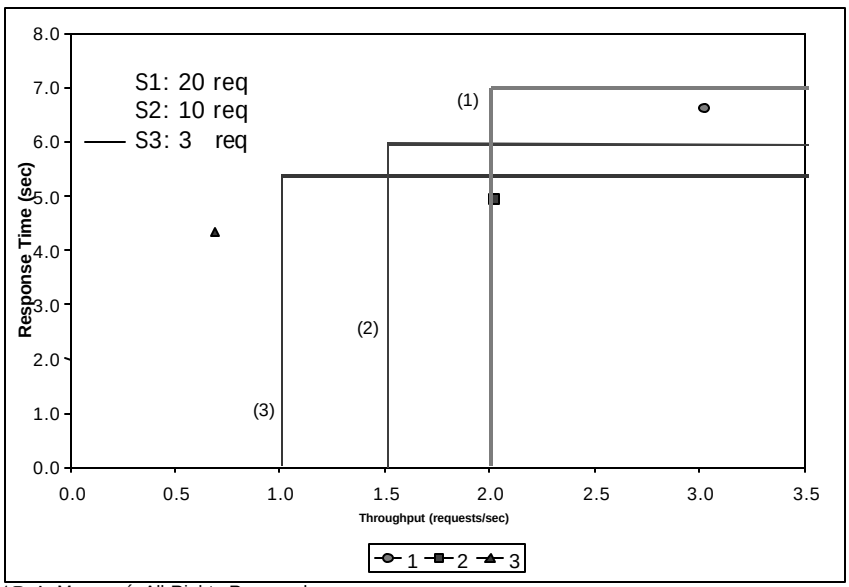
© 2004 D. A. Menascé. All Rights Reserved

Scenario 2: not feasible



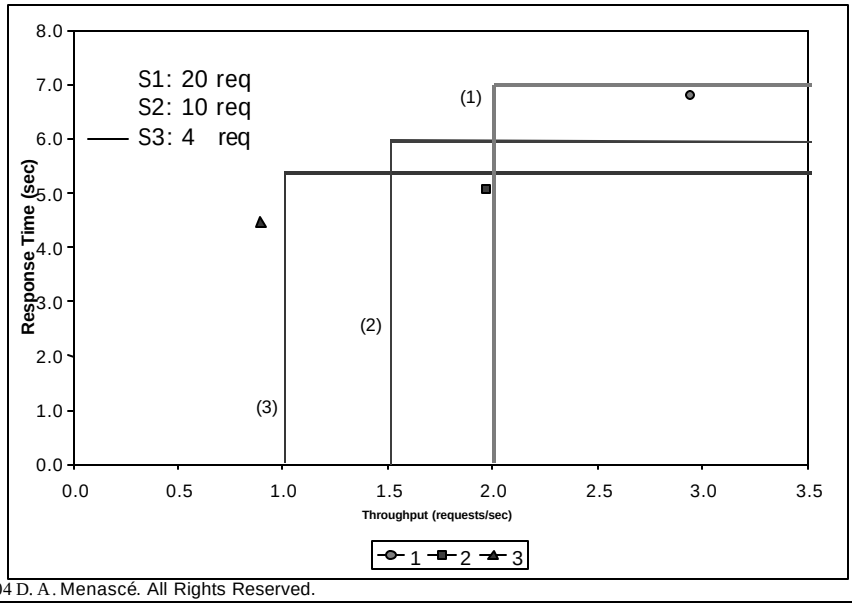
© 2004 D. A. Menascé. All Rights Reserved.

Scenario 3: not feasible



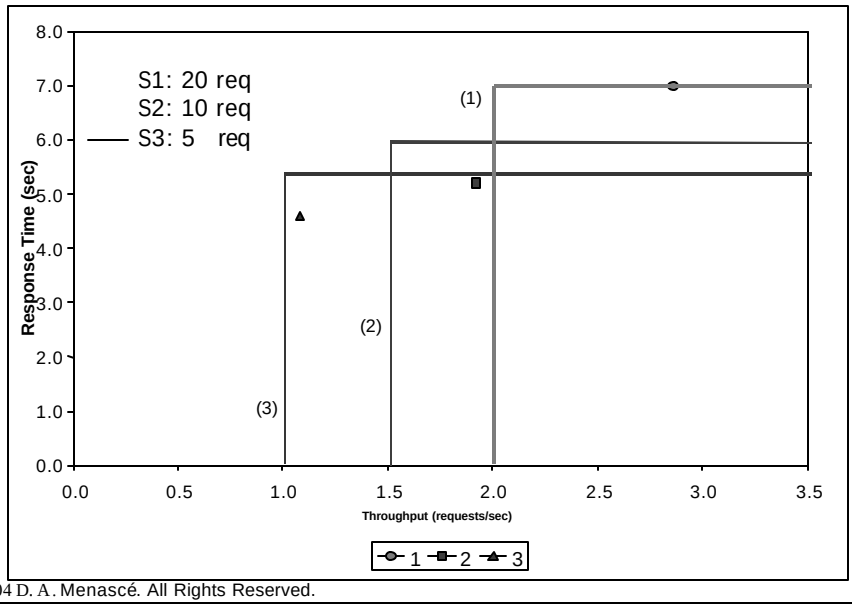
© 2004 D. A. Menascé. All Rights Reserved.

Scenario 4: not feasible



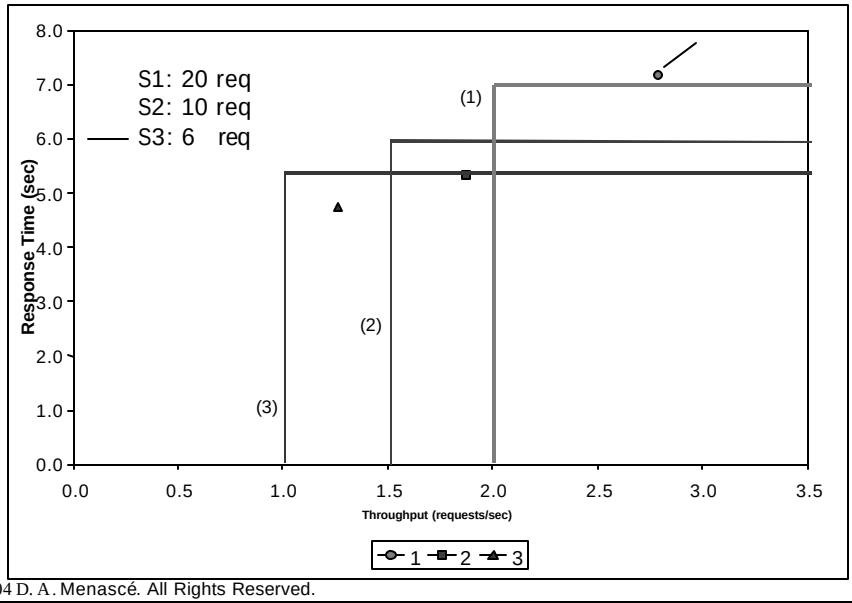
© 2004 D. A. Menascé. All Rights Reserved.

Scenario 5: feasible!



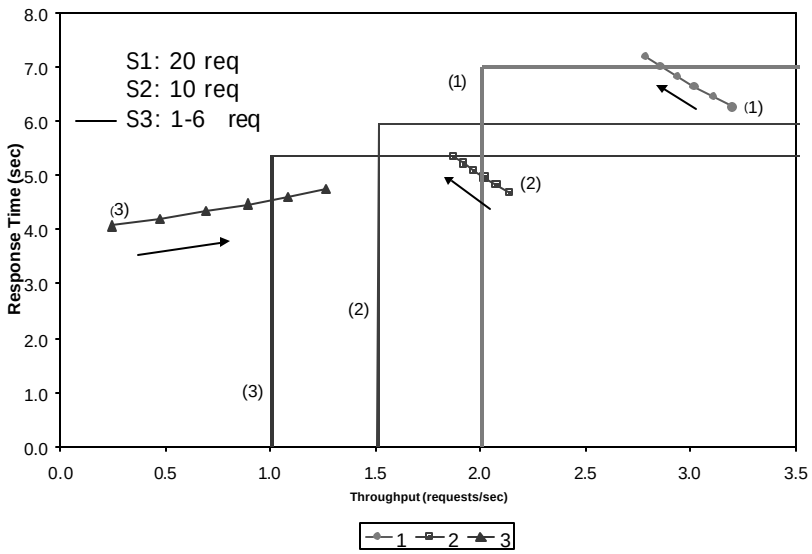
© 2004 D. A. Menascé. All Rights Reserved.

Scenario 6: not feasible again



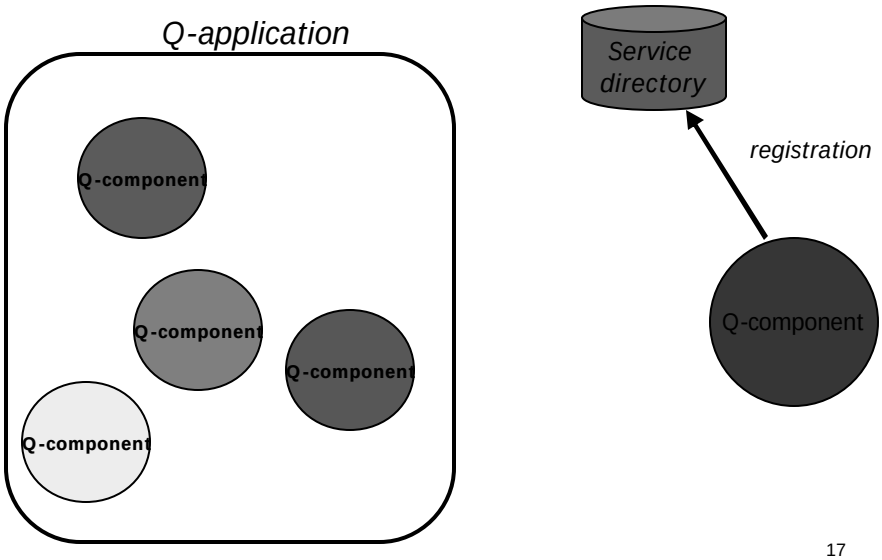
© 2004 D. A. Menascé. All Rights Reserved.

All six scenarios



© 2004 D. A. Menascé. All Rights Reserved.

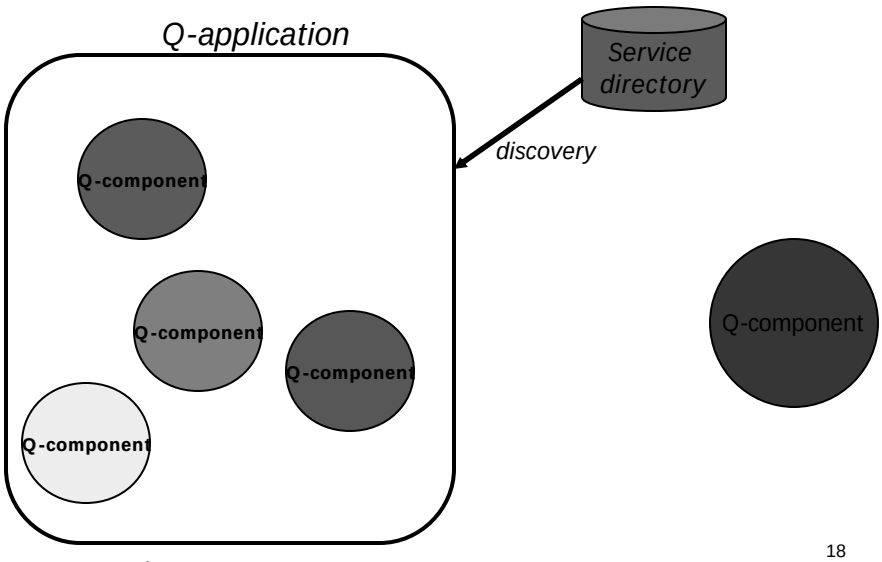
Q-Applications and Q-components



17

© 2004 D. A. Menascé. All Rights Reserved.

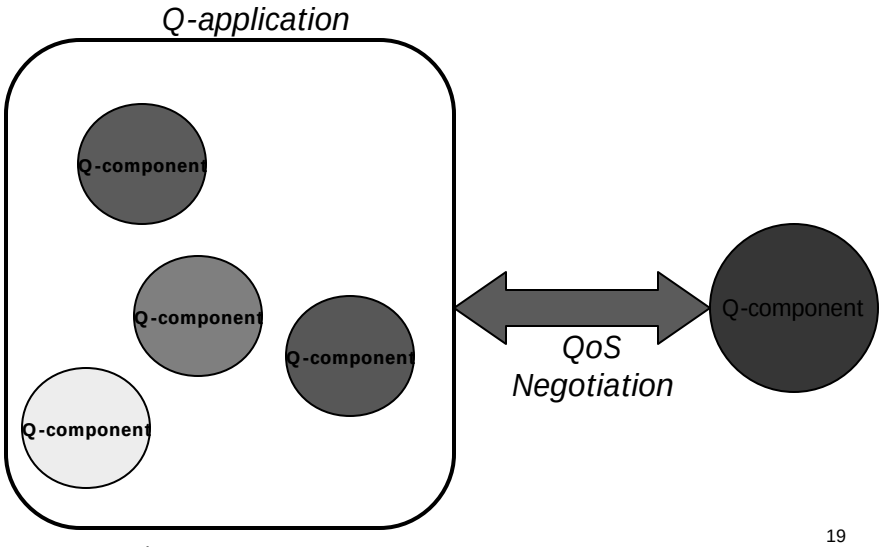
Q-Applications and Q-components



18

© 2004 D. A. Menascé. All Rights Reserved.

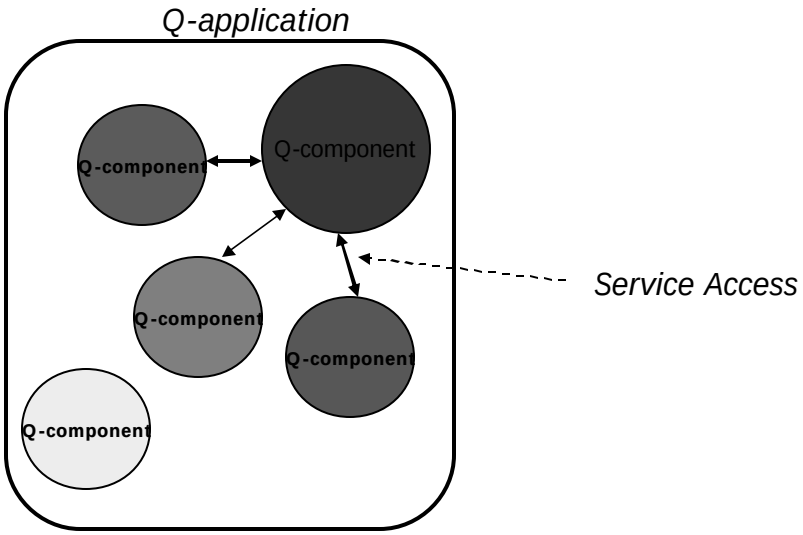
Q-Applications and Q-components



19

© 2004 D. A. Menascé. All Rights Reserved.

Q-Applications and Q-components



20

© 2004 D. A. Menascé. All Rights Reserved.

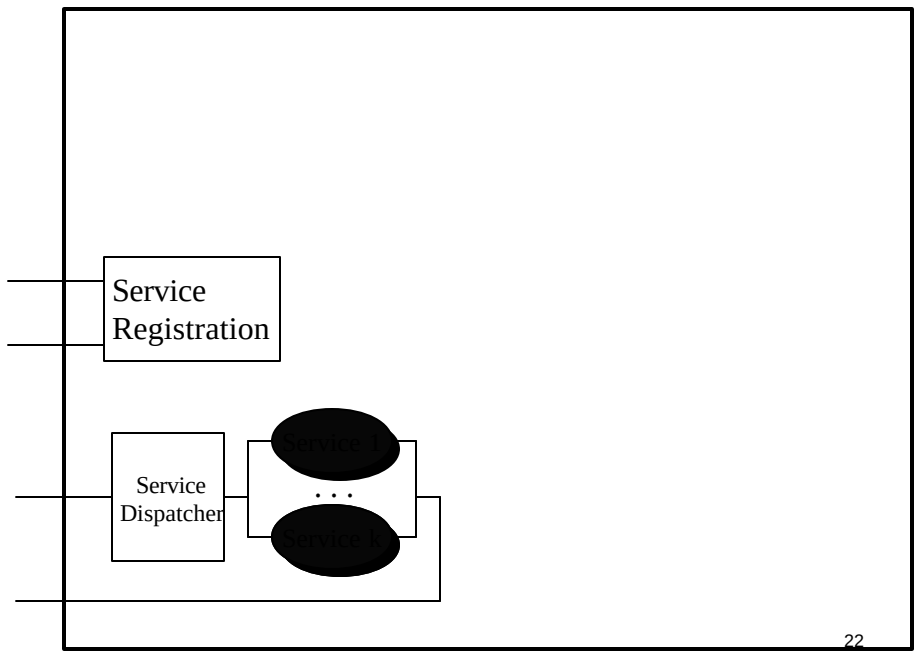
QoS-Aware Software Components: Q-Components

- Engage in QoS Negotiations (accept, reject, counter-offer)
- Provide QoS guarantees for multiple concurrent services
- Maintain a table of QoS commitments
- Service dispatching based on accepted QoS commitments
- Q-components are the building blocks of QoS-aware applications

21

© 2004 D. A. Menascé. All Rights Reserved.

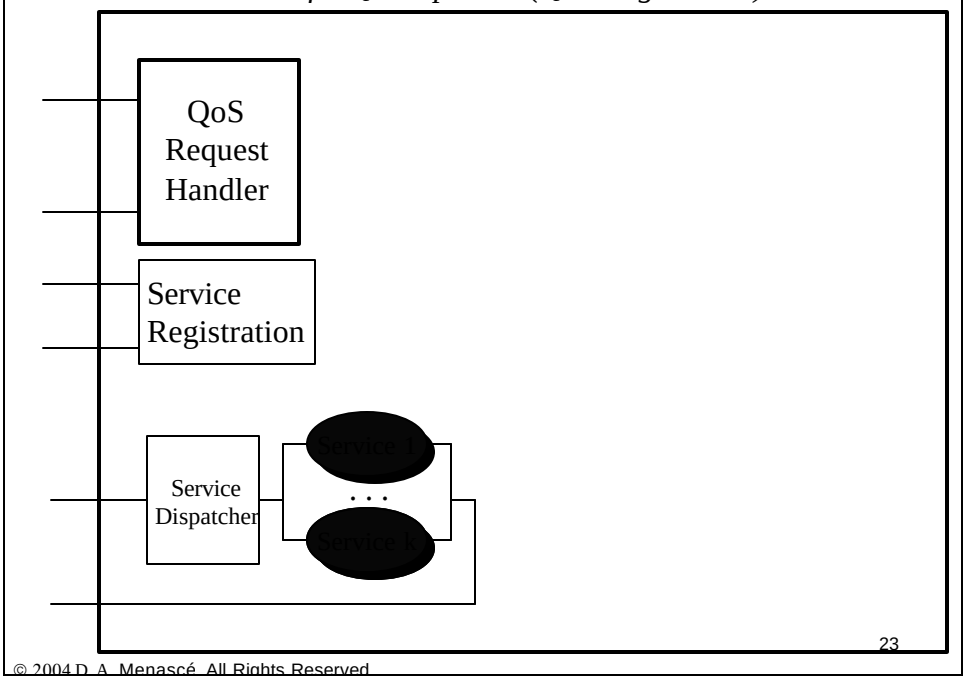
Architecture of a typical software component



22

© 2004 D. A. Menascé. All Rights Reserved.

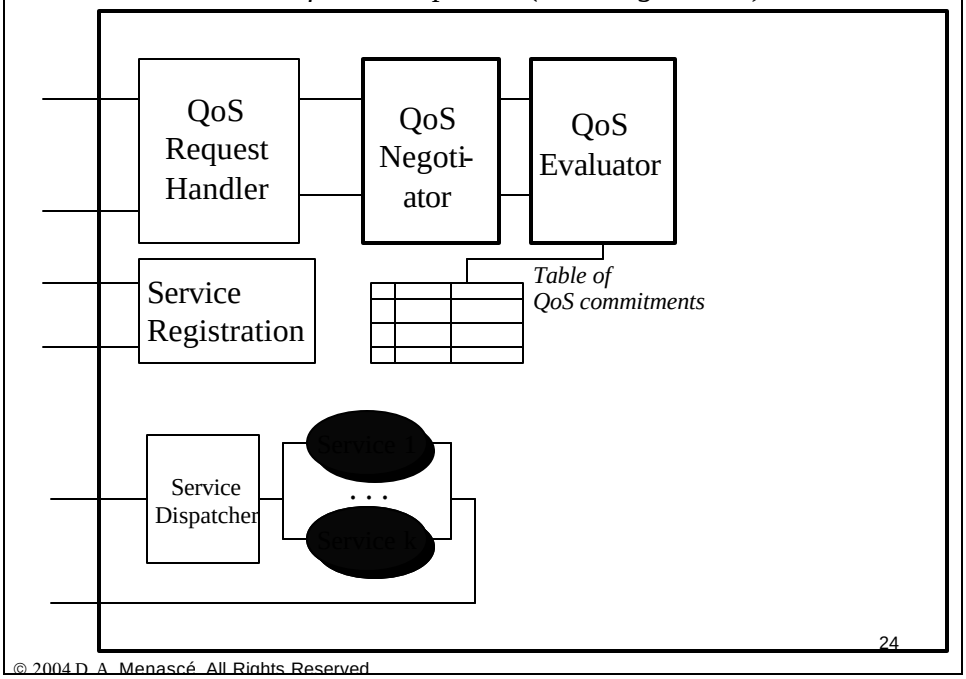
Architecture of a Q-component (QoS Negotiation)



23

© 2004 D. A. Menascé. All Rights Reserved

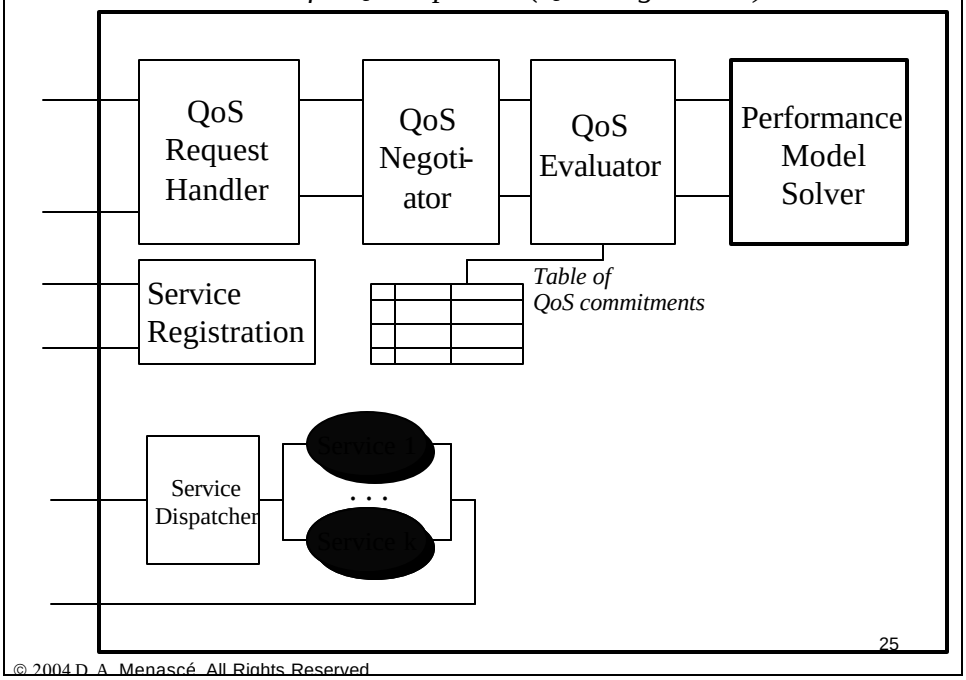
Architecture of a Q-component (QoS Negotiation)



24

© 2004 D. A. Menascé. All Rights Reserved

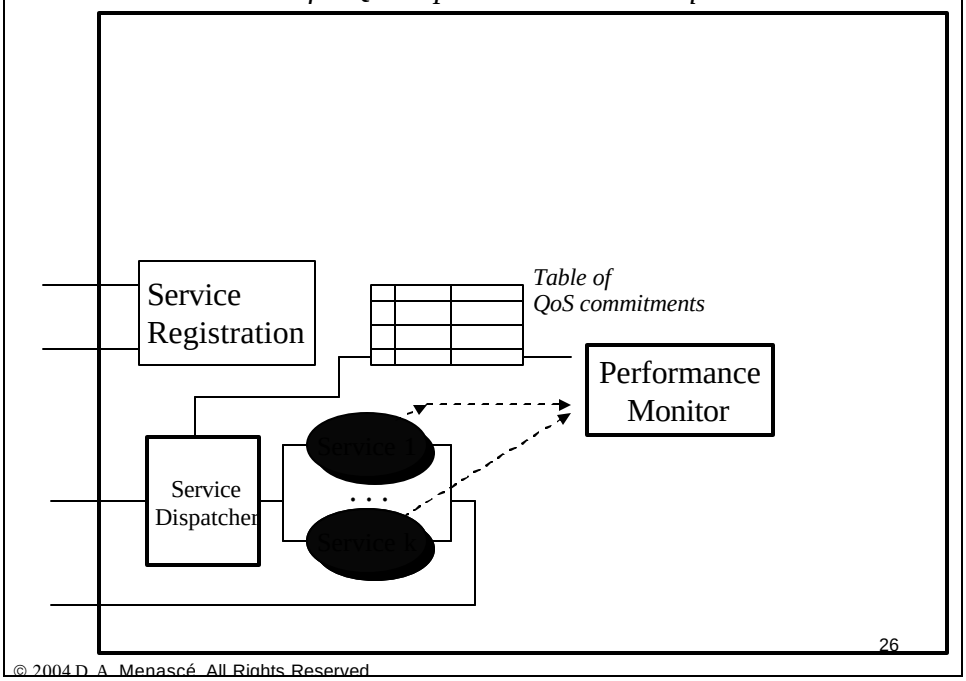
Architecture of a Q-component (QoS Negotiation)



25

© 2004 D. A. Menascé. All Rights Reserved

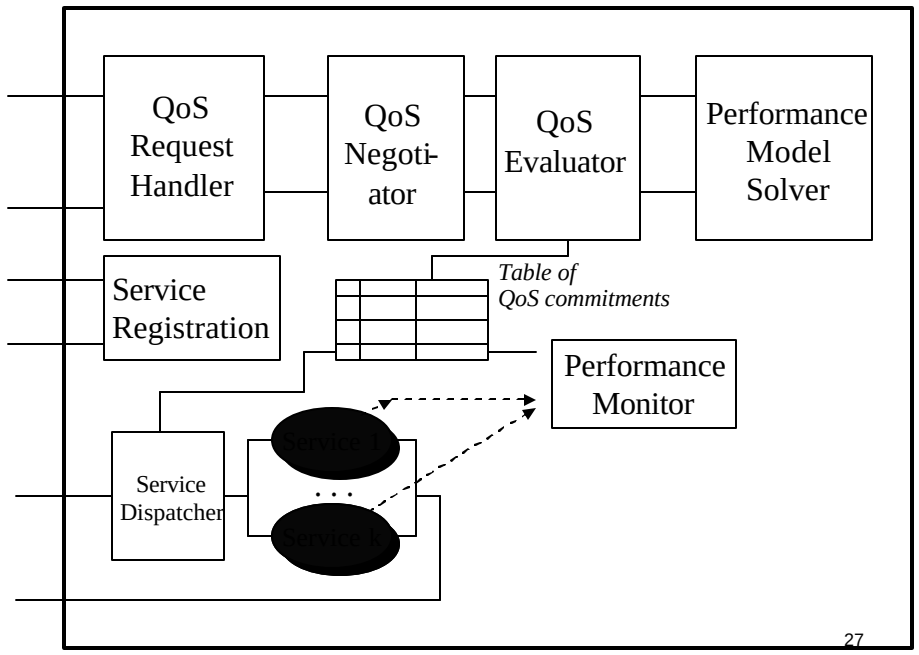
Architecture of a Q-component –Service Requests



26

© 2004 D. A. Menascé. All Rights Reserved

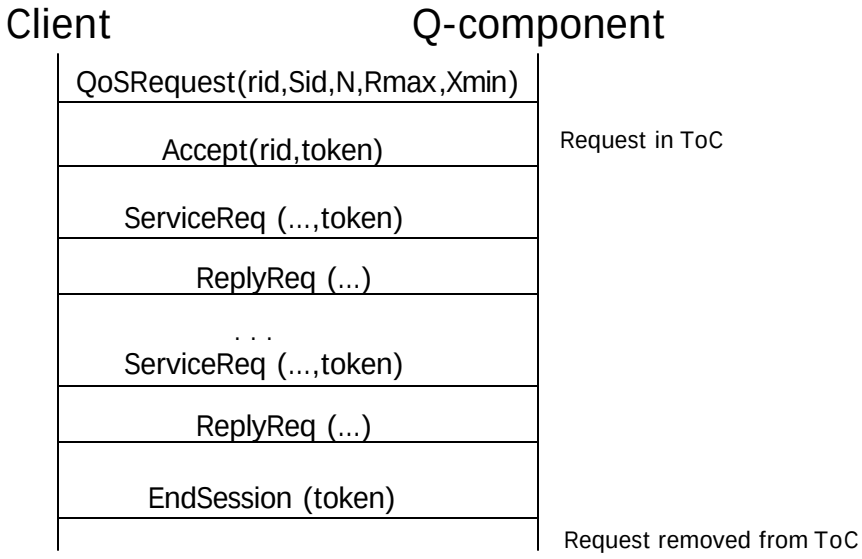
Architecture of a Q-component



27

© 2004 D. A. Menascé. All Rights Reserved.

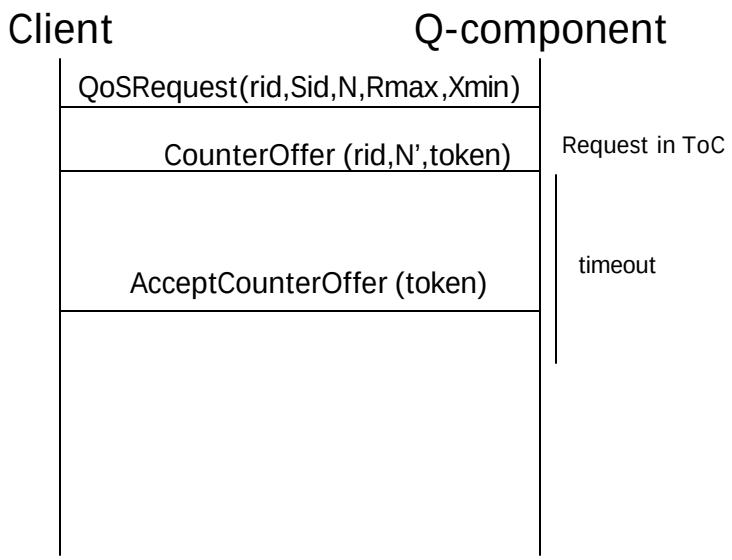
Successful QoS Negotiation



28

© 2004 D. A. Menascé. All Rights Reserved.

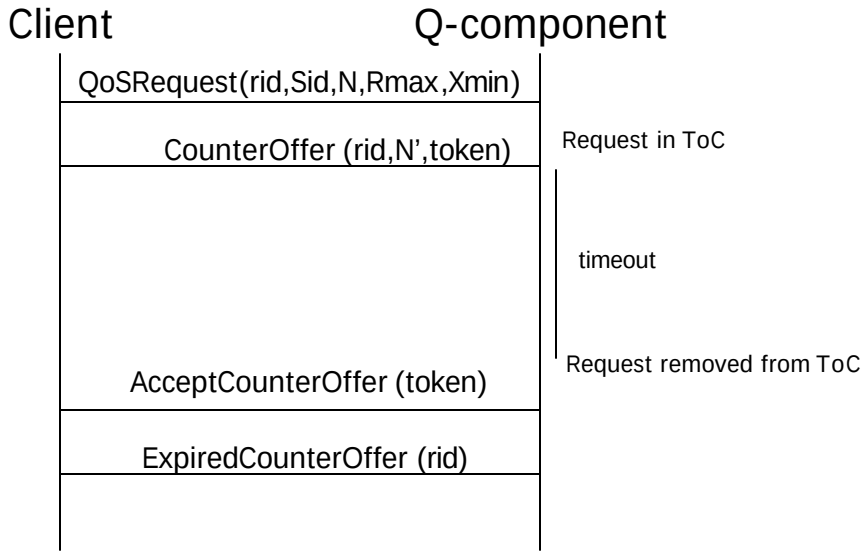
On-time Accepted Counteroffer



29

© 2004 D. A. Menascé. All Rights Reserved.

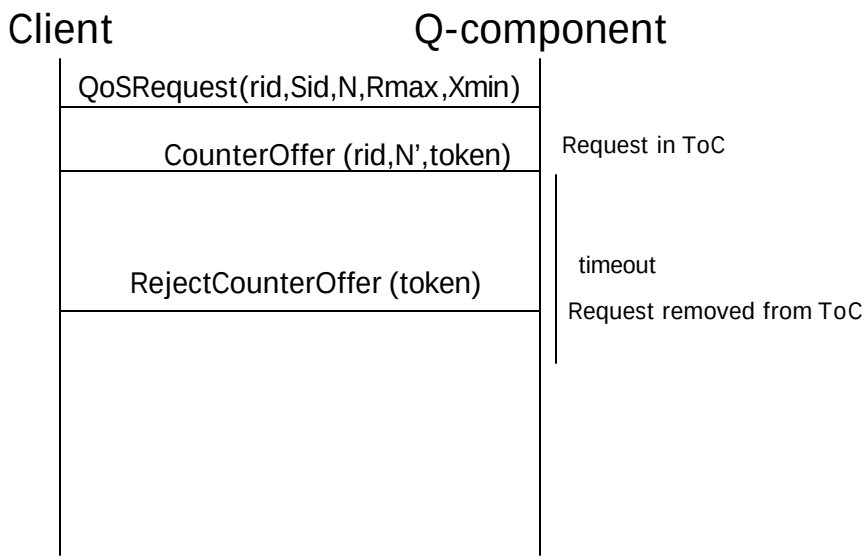
Expired Accepted Counteroffer



30

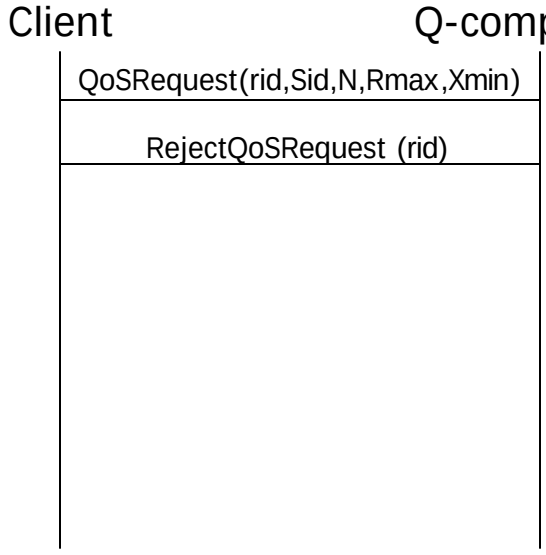
© 2004 D. A. Menascé. All Rights Reserved.

Rejected Counteroffer



© 2004 D. A. Menascé. All Rights Reserved.

Rejected QoS Negotiation



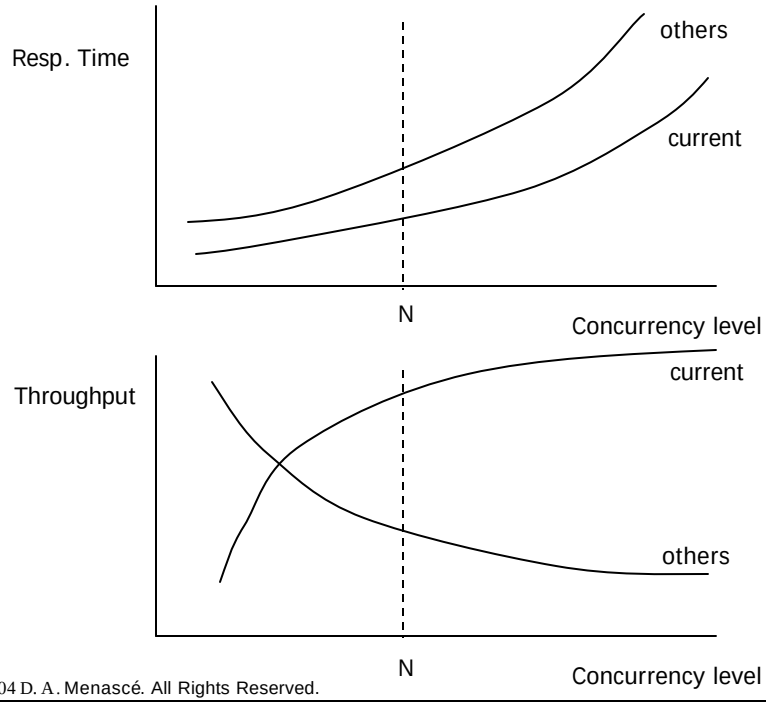
© 2004 D. A. Menascé. All Rights Reserved.

Decision Table for QoS Negotiation

1. Current and other requests are satisfied				Accept
2. Only Current is Violated				
Reason	Remedy	Current	Others	Decision
Only MAXR is violated	Decrease N	OK	OK	Counter Offer
		Not OK: MINX is violated or N=0	OK	Reject
Only MINX is violated	Increase N	OK	OK	Counter Offer
		OK	Not OK	Reject
		Not OK & MAXR is violated	OK	Reject
		Not OK & MAXR is violated	Not OK	Reject
MINX & MAXR are violated	Decreasing N reduces X and increasing N increases R. So, there is no solution.			Reject
3. Current Request and Others are Violated				
Reason	Remedy	Current	Others	Decision
Only MAXR is violated	Decrease N	OK	OK	Counter Offer
		OK	Not OK	Reject
		Not OK: MINX is violated or N=0	OK or not OK	Reject
Only MINX is violated	X could be increased by increasing N. But this would further violate the QoS of other classes.			Reject
MINX & MAXR are violated	Decreasing N reduces X and increasing N increases R. So, there is no solution.			Reject
4. Only Other Requests are Violated				
Reason	Remedy	Current	Others	Decision
Any	Decrease N	OK	OK	Counter Offer
		OK	Not OK: N=1 but others still violated	Reject
		Not OK: MINX violated or N=0	OK or not OK	Reject

© 2004 D. A. Menascé. All Rights Reserved.

33



© 2004 D. A. Menascé. All Rights Reserved.

34

Building a Performance Model

New Request: Sid = 3, N = 12

Base Matrix of Service Demands (in msec):

	Service		
	1	2	3
CPU	25	34	20
Disk 1	30	50	24
Disk 2	28	42	31

Table of Commitments (ToC):

Commitment ID	Service ID	N	…
1	2	10	…
2	3	15	…
3	1	8	…
4	1	20	…
5	2	13	…

Matrix of Service Demands (in msec)

	Class					
	1	2	3	4	5	6
CPU	34	20	25	25	34	20
Disk 1	50	24	30	30	50	24
Disk 2	42	31	28	28	42	31
Vector N:	10	15	8	20	13	12

© 2004 D. A. Menascé. All Rights Reserved.

Building a Performance Model

New Request: Sid = 3, N = 12

Base Matrix of Service Demands (in msec):

	Service		
	1	2	3
CPU	25	34	20
Disk 1	30	50	24
Disk 2	28	42	31

Table of Commitments (ToC):

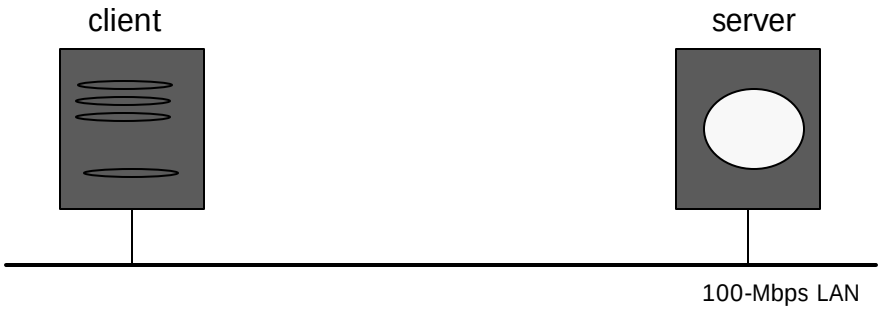
Commitment ID	Service ID	N	…
1	2	10	…
2	3	15	…
3	1	8	…
4	1	20	…
5	2	13	…

Matrix of Service Demands (in msec)

	Class					
	1	2	3	4	5	6
CPU	34	20	25	25	34	20
Disk 1	50	24	30	30	50	24
Disk 2	42	31	28	28	42	31
Vector N:	10	15	8	20	13	12

© 2004 D. A. Menascé. All Rights Reserved.

Experimental Setup



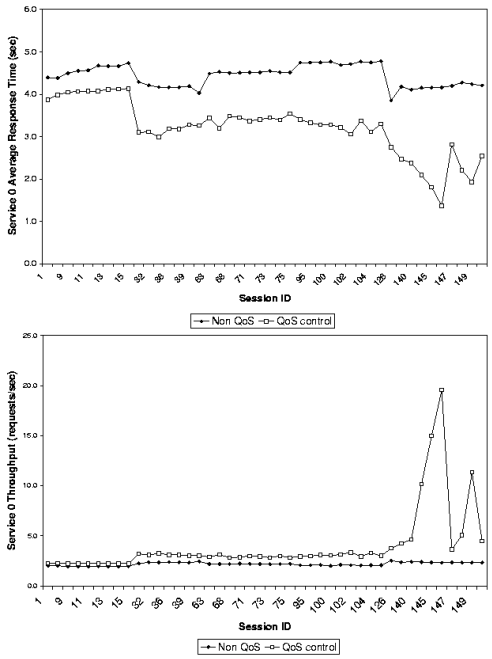
- 15 client processes.
- Each client process generates 20 sessions.
- Each session forks 10 threads.
- Each thread sends 5 service requests.

- Q-component.
- 3 services

A random workload is generated and submitted to a non-QoS component. The workload is recorded and replayed against a Q-component.

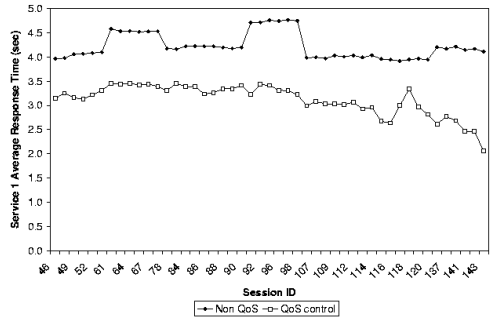
© 2004 D. A. Menascé. All Rights Reserved.

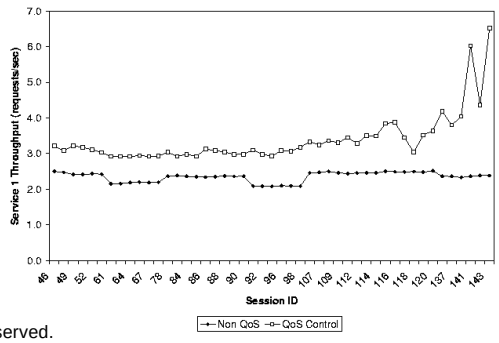
Service 0 Results:



© 2004 D. A. Menascé. All Rights Reserved.

Service 1 Results:

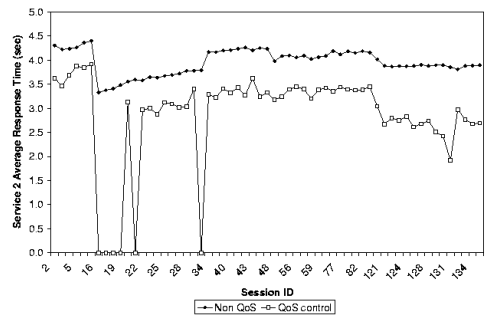


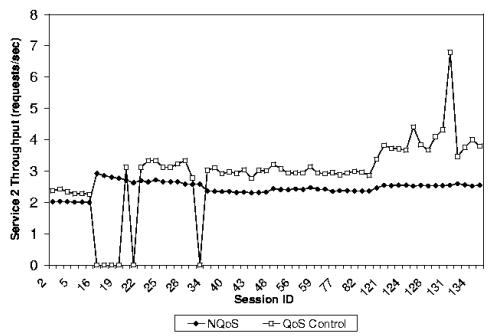


© 2004 D. A. Menascé. All Rights Reserved.

39

Service 2 Results:





© 2004 D. A. Menascé. All Rights Reserved.

40

f = 0.0				
Svc No.	No. Dropped Sessions	No. of Sessions	% Drop	% Resp. Time Reduction
0	24	440	5	11
1	19	470	4	9
2	59	590	10	7
Total	102	1500	7	9
f = 0.10				
Svc No.	No. Dropped Sessions	No. of Sessions	% Drop	% Resp. Time Reduction
0	52	440	12	21
1	66	470	14	16
2	148	590	25	12
Total	266	1500	18	16
f = 0.25				
Svc No.	No. Dropped Sessions	No. of Sessions	% Drop	% Resp. Time Reduction
0	92	440	21	28
1	140	470	30	26
2	263	590	45	20
Total	102	1500	33	24

41

© 2004 D. A. Menascé. All Rights Reserved.

Concluding remarks

- A validated framework for QoS-aware software components that do admission control and resource reservation.
- Analytic performance models can be very useful and efficient in QoS negotiation.
- QoS negotiation overhead did not exceed 10% of the CPU service demand.

42

Ongoing Work

- Self-configurable component based software:
  - Experiments with different QoS negotiation approaches
  - QoS-aware applications
  - Include cost in the QoS negotiation
  - Case of components invoking other components.

43

© 2004 D. A. Menascé. All Rights Reserved.

?

www.cs.gmu.edu/faculty/menasce.html

44

© 2003 D. A. Menascé. All Rights Reserved.