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

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.
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.

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

Scenario 1: not feasible

© 2004 D. A. Menascé. All Rights Reserved.
Scenario 2: not feasible

Scenario 3: not feasible
Scenario 4: not feasible

Scenario 5: feasible!
Scenario 6: not feasible again

All six scenarios
Q-Applications and Q-components

Q-application

Q-component

Q-component

Q-component

Service directory

registration

Q-component

Q-Applications and Q-components

Q-application

Q-component

Q-component

Q-component

Q-component

Q-component

Service directory

discovery

Q-component

Q-component

© 2004 D. A. Menascé. All Rights Reserved.
Q-Applications and Q-components

Q-application

Q-component

Q-component

Q-component

Q-component

QoS Negotiation

© 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

Architecture of a typical software component
Architecture of a Q-component (QoS Negotiation)

QoS Request Handler

Service Registration

Service Dispatcher

QoS Negotiator

QoS Evaluator

Table of QoS commitments

© 2004 D. A. Menascé. All Rights Reserved.
Architecture of a Q-component (QoS Negotiation)

- QoS Request Handler
- QoS Negotiator
- QoS Evaluator
- Performance Model Solver
- Service Registration
- Service Dispatcher

Table of QoS commitments

Architecture of a Q-component – Service Requests

- Service Registration
- Service Dispatcher
- Performance Monitor

Table of QoS commitments

© 2004 D. A. Menascé. All Rights Reserved.
Successful QoS Negotiation

<table>
<thead>
<tr>
<th>Client</th>
<th>Q-component</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoSRequest(rid,Sid,N,Rmax,Xmin)</td>
<td></td>
</tr>
<tr>
<td>Accept(rid,token)</td>
<td>Request in ToC</td>
</tr>
<tr>
<td>ServiceReq (...token)</td>
<td></td>
</tr>
<tr>
<td>ReplyReq (...)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Request removed from ToC</td>
</tr>
<tr>
<td>ServiceReq (...token)</td>
<td></td>
</tr>
<tr>
<td>ReplyReq (...)</td>
<td></td>
</tr>
<tr>
<td>EndSession (token)</td>
<td></td>
</tr>
</tbody>
</table>
On-time Accepted Counteroffer

Client

QoSRequest(rid, Sid, N, Rmax, Xmin)

CounterOffer (rid, N', token)

AcceptCounterOffer (token)

Q-component

Request in ToC

timeout

Expired Accepted Counteroffer

Client

QoSRequest(rid, Sid, N, Rmax, Xmin)

CounterOffer (rid, N', token)

AcceptCounterOffer (token)

ExpiredCounterOffer (rid)

Q-component

Request in ToC

timeout

Request removed from ToC

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

<table>
<thead>
<tr>
<th>Client</th>
<th>Q-component</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoSRequest(rid,Sid,N,Rmax,Xmin)</td>
<td>Request in ToC</td>
</tr>
<tr>
<td>CounterOffer (rid,N',token)</td>
<td>timeout</td>
</tr>
<tr>
<td>RejectCounterOffer (token)</td>
<td>Request removed from ToC</td>
</tr>
</tbody>
</table>

Rejected QoS Negotiation

<table>
<thead>
<tr>
<th>Client</th>
<th>Q-component</th>
</tr>
</thead>
<tbody>
<tr>
<td>QoSRequest(rid,Sid,N,Rmax,Xmin)</td>
<td>RejectQoSRequest (rid)</td>
</tr>
</tbody>
</table>
## Decision Table for QoS Negotiation

<table>
<thead>
<tr>
<th>Reason</th>
<th>Remedy</th>
<th>Current</th>
<th>Others</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only MAXR is violated</td>
<td>Decrease N</td>
<td>OK</td>
<td>OK</td>
<td>Counter Offer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OK</td>
<td>OK</td>
<td>Reject</td>
</tr>
<tr>
<td></td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>Counter Offer</td>
</tr>
<tr>
<td></td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>Reject</td>
</tr>
<tr>
<td></td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>Counter Offer</td>
</tr>
<tr>
<td></td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>Reject</td>
</tr>
<tr>
<td>Only MINX is violated</td>
<td>Increase N</td>
<td>OK</td>
<td>OK</td>
<td>Counter Offer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OK</td>
<td>OK</td>
<td>Reject</td>
</tr>
<tr>
<td></td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>Counter Offer</td>
</tr>
<tr>
<td></td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>Reject</td>
</tr>
</tbody>
</table>

**5. Current and other requests are satisfied**

- **Accept**
  - OK
  - OK
  - Counter Offer

- **Reject**
  - OK
  - OK
  - Counter Offer

- **Not OK**
  - OK
  - OK
  - Counter Offer

**Reasons for Violation**

- **MAXR is violated**
  - Decreasing N reduces R, and increasing N increases R. So, there is no solution.
  - Reject

- **MINX is violated**
  - Increasing X reduces N, and increasing N increases X. So, there is no solution.
  - Reject

- **Both MINX and MAXR are violated**
  - Decreasing N reduces N, and increasing N increases R. So, there is no solution.
  - Reject

**Cases**

1. **Current and other requests are satisfied**
   - Accept
   - OK
   - OK
   - Counter Offer
   - Reject

2. **Only Current is Violated**
   - Accept
   - OK
   - OK
   - Counter Offer
   - Reject

3. **Only MINX is violated**
   - Accept
   - OK
   - OK
   - Counter Offer
   - Reject

4. **Only MAXR is violated**
   - Accept
   - OK
   - OK
   - Counter Offer
   - Reject

5. **Both MINX and MAXR are violated**
   - Accept
   - OK
   - OK
   - Counter Offer
   - Reject

**Decision**

- **Accept**
  - OK
  - OK
  - Counter Offer

- **Reject**
  - OK
  - OK
  - Counter Offer

**Notes**

- Decreasing N reduces X and increasing N increases R. So, there is no solution.
- Increasing X reduces N, and increasing N increases X. So, there is no solution.

**Graphs**

- **Resp. Time**
  - **current**
  - **others**

- **Concurrency level**
  - **current**
  - **others**

- **Throughput**
  - **current**
  - **others**

© 2004 D. A. Menascé. All Rights Reserved.
# Building a Performance Model

**New Request:** Sid = 3, N = 12

## Base Matrix of Service Demands (in msec):

<table>
<thead>
<tr>
<th>Service</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>25</td>
<td>34</td>
<td>20</td>
</tr>
<tr>
<td>Disk 1</td>
<td>30</td>
<td>50</td>
<td>24</td>
</tr>
<tr>
<td>Disk 2</td>
<td>28</td>
<td>42</td>
<td>31</td>
</tr>
</tbody>
</table>

## Matrix of Service Demands (in msec)

<table>
<thead>
<tr>
<th>Class</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>34</td>
<td>20</td>
<td>25</td>
<td>25</td>
<td>34</td>
<td>20</td>
</tr>
<tr>
<td>Disk 1</td>
<td>50</td>
<td>24</td>
<td>30</td>
<td>30</td>
<td>50</td>
<td>24</td>
</tr>
<tr>
<td>Disk 2</td>
<td>42</td>
<td>31</td>
<td>28</td>
<td>28</td>
<td>42</td>
<td>31</td>
</tr>
</tbody>
</table>

## Vector N:

| 10 | 15 | 8  | 20 | 13 | 12 |

---

# Building a Performance Model

**New Request:** Sid = 3, N = 12

## Base Matrix of Service Demands (in msec):

<table>
<thead>
<tr>
<th>Service</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>25</td>
<td>34</td>
<td>20</td>
</tr>
<tr>
<td>Disk 1</td>
<td>30</td>
<td>50</td>
<td>24</td>
</tr>
<tr>
<td>Disk 2</td>
<td>28</td>
<td>42</td>
<td>31</td>
</tr>
</tbody>
</table>

## Matrix of Service Demands (in msec)

<table>
<thead>
<tr>
<th>Class</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>34</td>
<td>20</td>
<td>25</td>
<td>25</td>
<td>34</td>
<td>20</td>
</tr>
<tr>
<td>Disk 1</td>
<td>50</td>
<td>24</td>
<td>30</td>
<td>30</td>
<td>50</td>
<td>24</td>
</tr>
<tr>
<td>Disk 2</td>
<td>42</td>
<td>31</td>
<td>28</td>
<td>28</td>
<td>42</td>
<td>31</td>
</tr>
</tbody>
</table>

## Vector N:

| 10 | 15 | 8  | 20 | 13 | 12 |
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.

Service 0
Results:
Service 1
Results:

Service 2
Results:
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.
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.

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