Achieving QoS in Complex Distributed Systems through Autonomic Computing

Dr. Daniel A. Menascé
Department of Computer Science
George Mason University
www.cs.gmu.edu/faculty/menasce.html

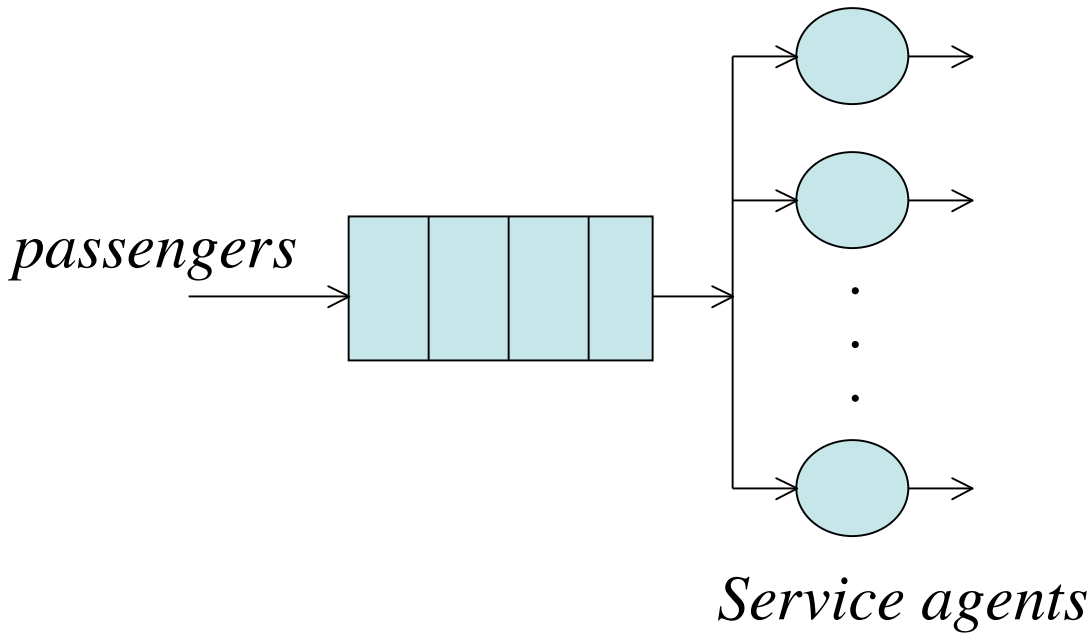
© 2005 Daniel A. Menascé. All Rights Reserved.

Approaches to QoS Management

- Overprovisioning: design for peak load.
  - Expensive!
  - Costs include hardware, software licenses, personnel, power, air conditioning, floor space, and communication.
  - What if you don't know what the peak load will be?
- Dynamic reconfiguration to adapt to changes in workload intensity.
  - Get the best QoS out of existing deployed capacity
  - Can delay upgrades and save money.

© 2005 Daniel A. Menascé. All Rights Reserved.

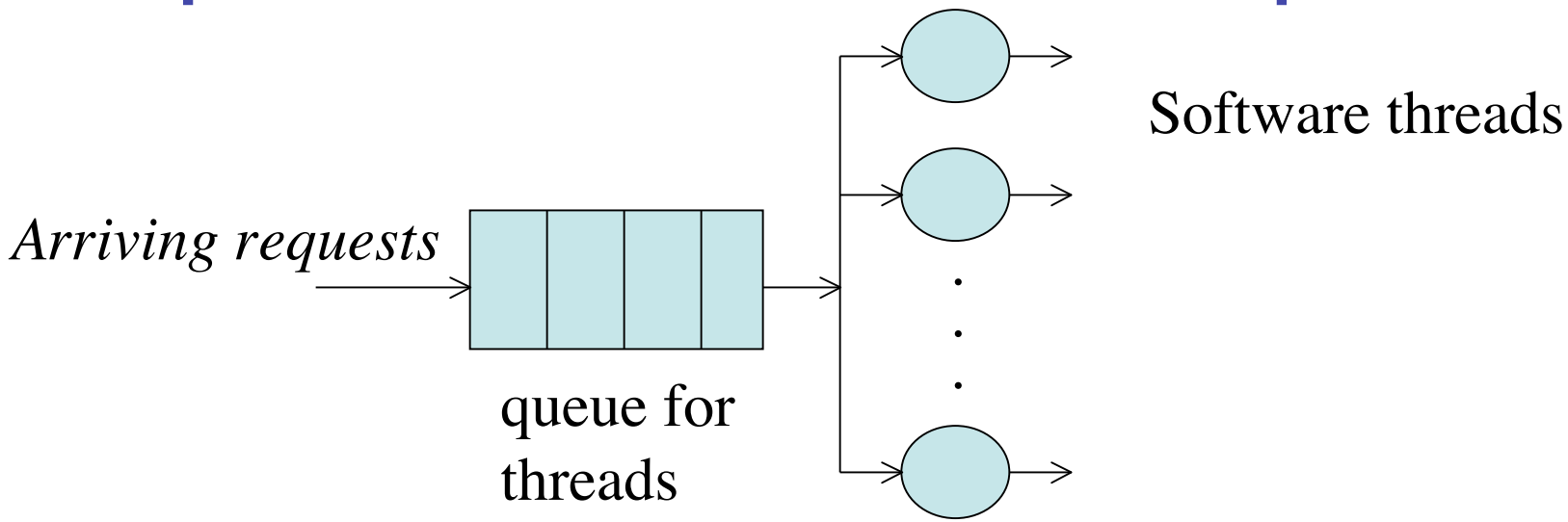
Simple motivational example Check-in Line at Airport



Q: How does the response time vary with the number of service agents?

© 2005 Daniel A. Menascé. All Rights Reserved.

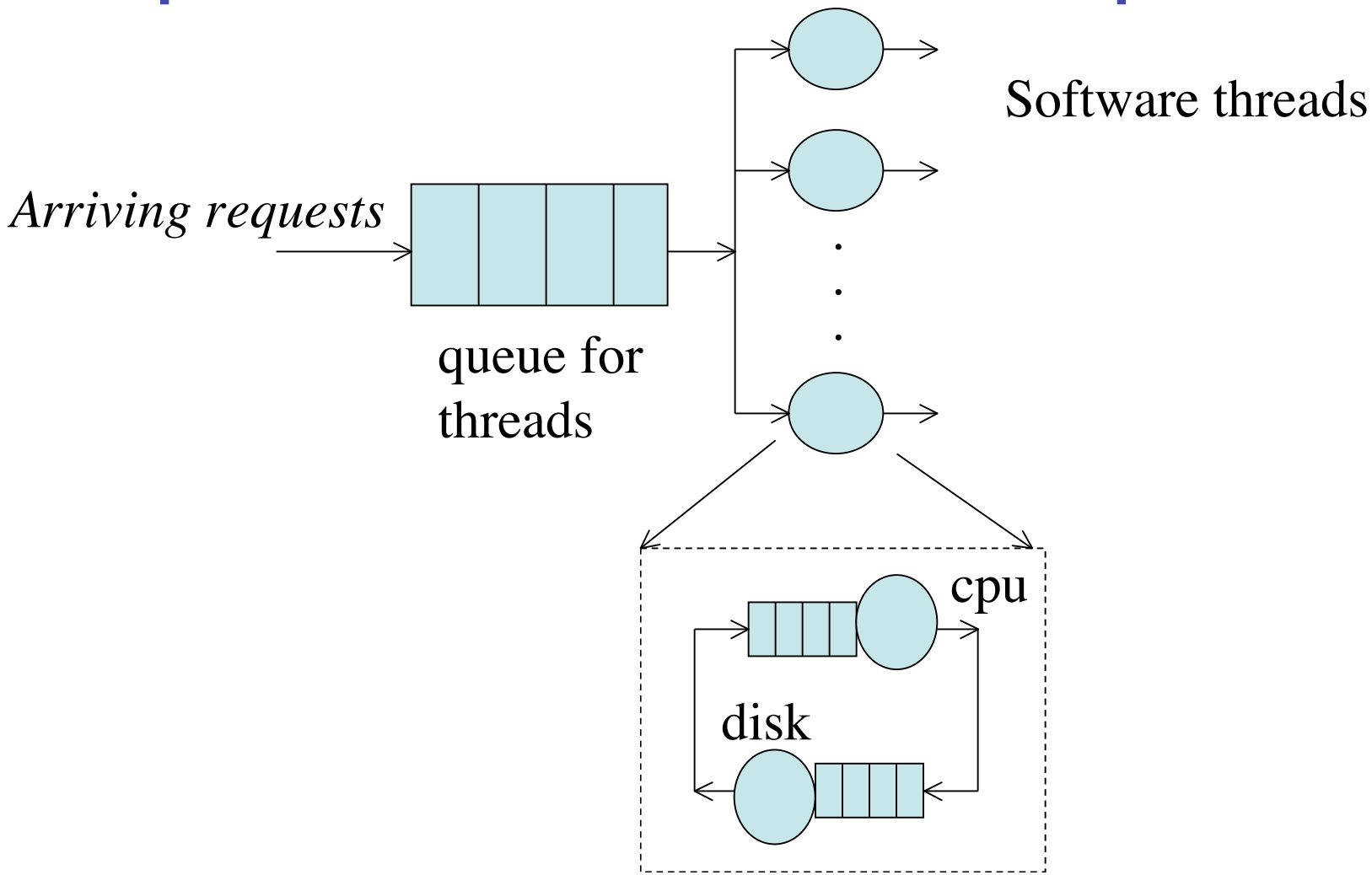
Simple motivational example



Q: How does the response time vary with the number of software threads?

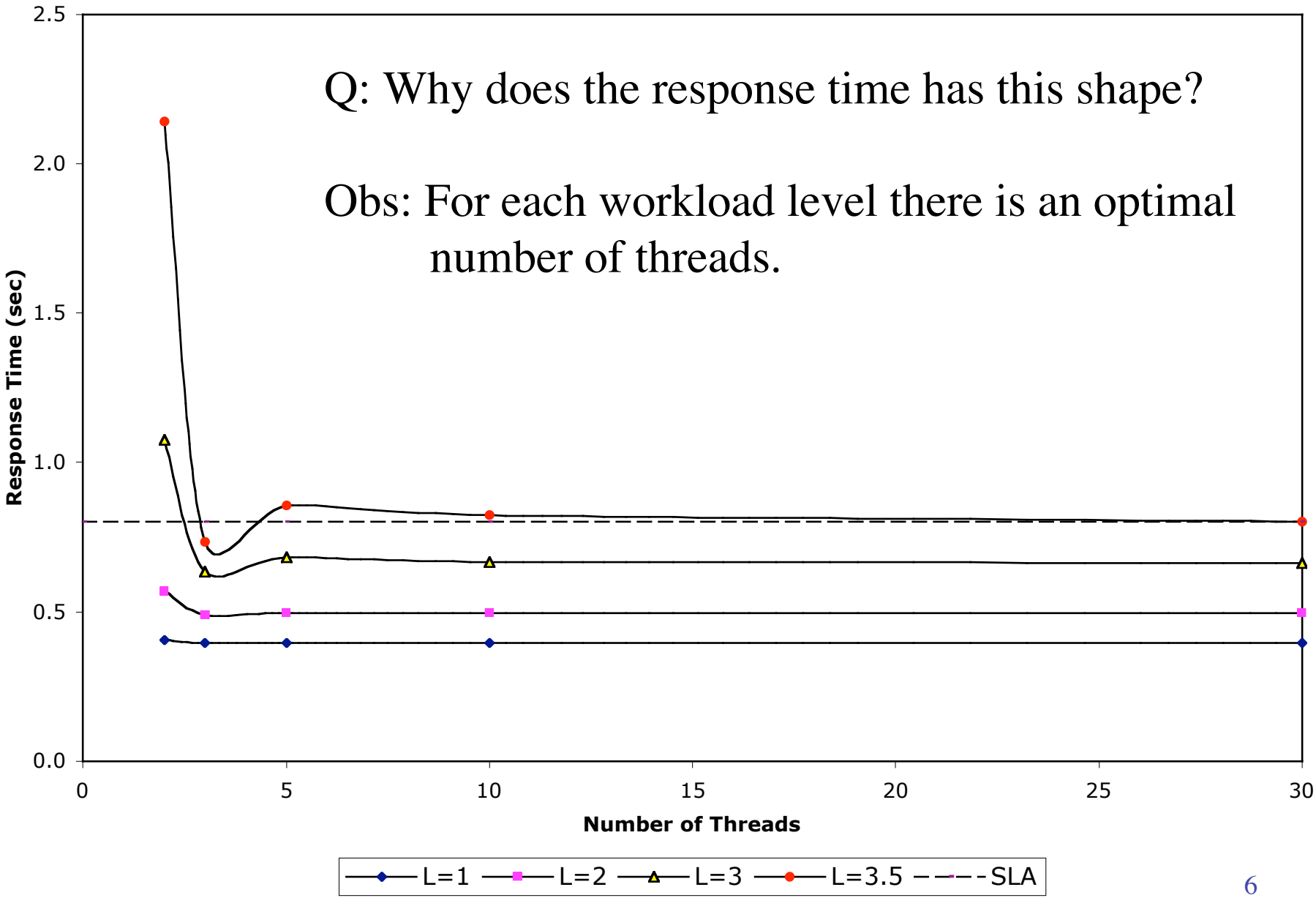
© 2005 Daniel A. Menascé. All Rights Reserved.

Simple motivational example



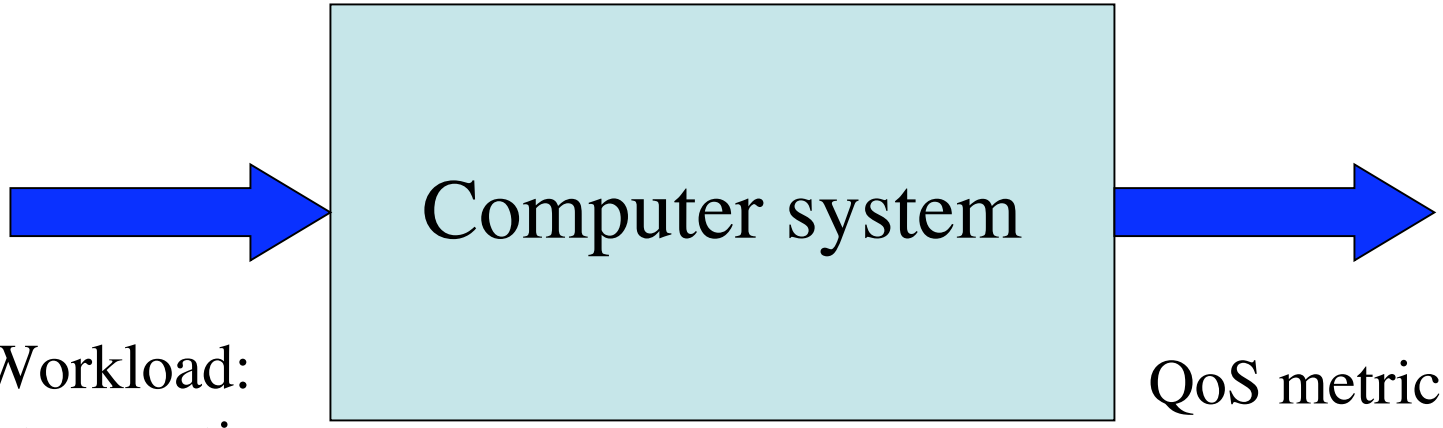
Q: How does the response time vary with the number of software threads?

© 2005 Daniel A. Menascé. All Rights Reserved.



© 2005 Daniel A. Menascé. All Rights Reserved.

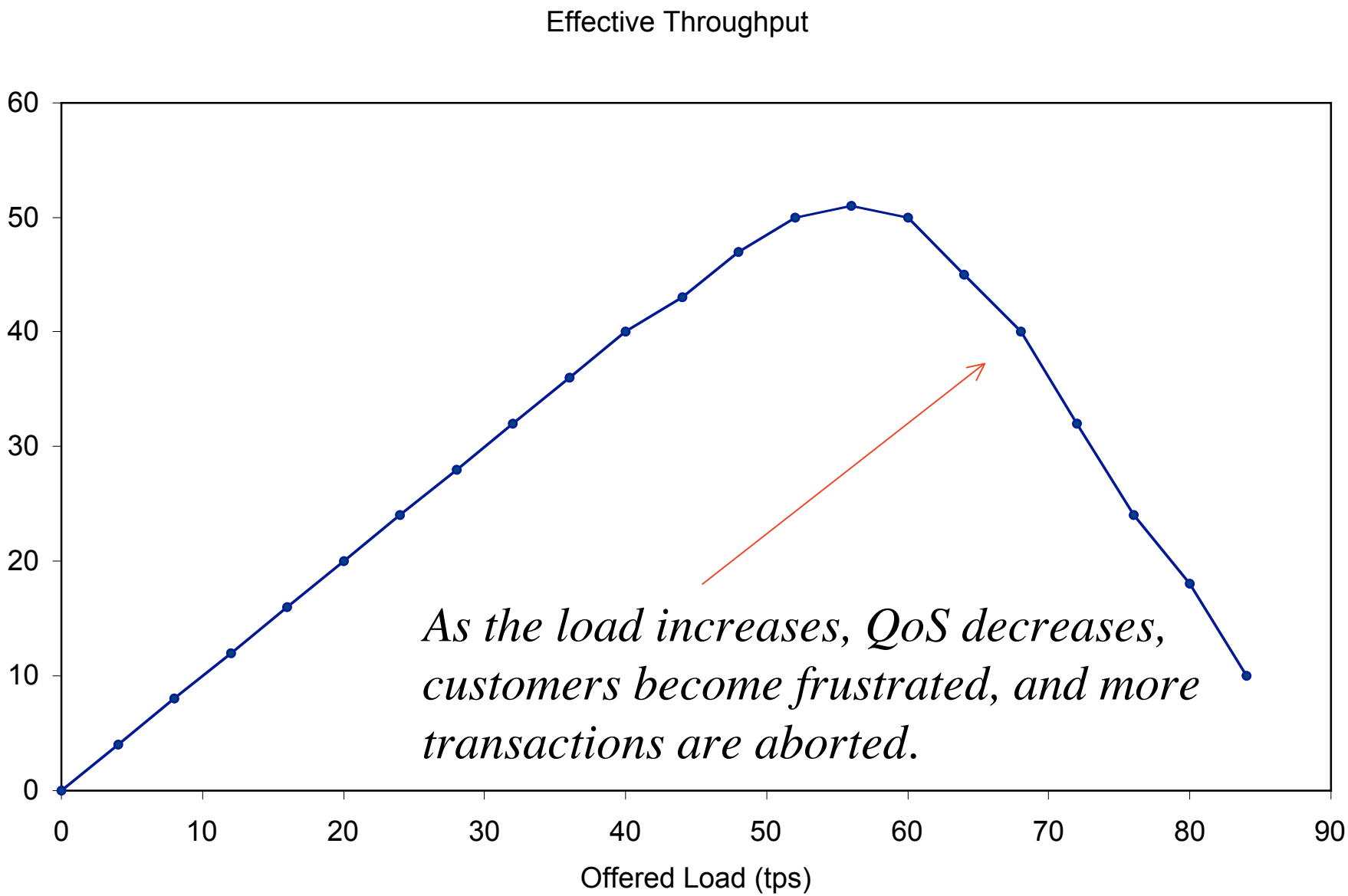
Workload and QoS metrics



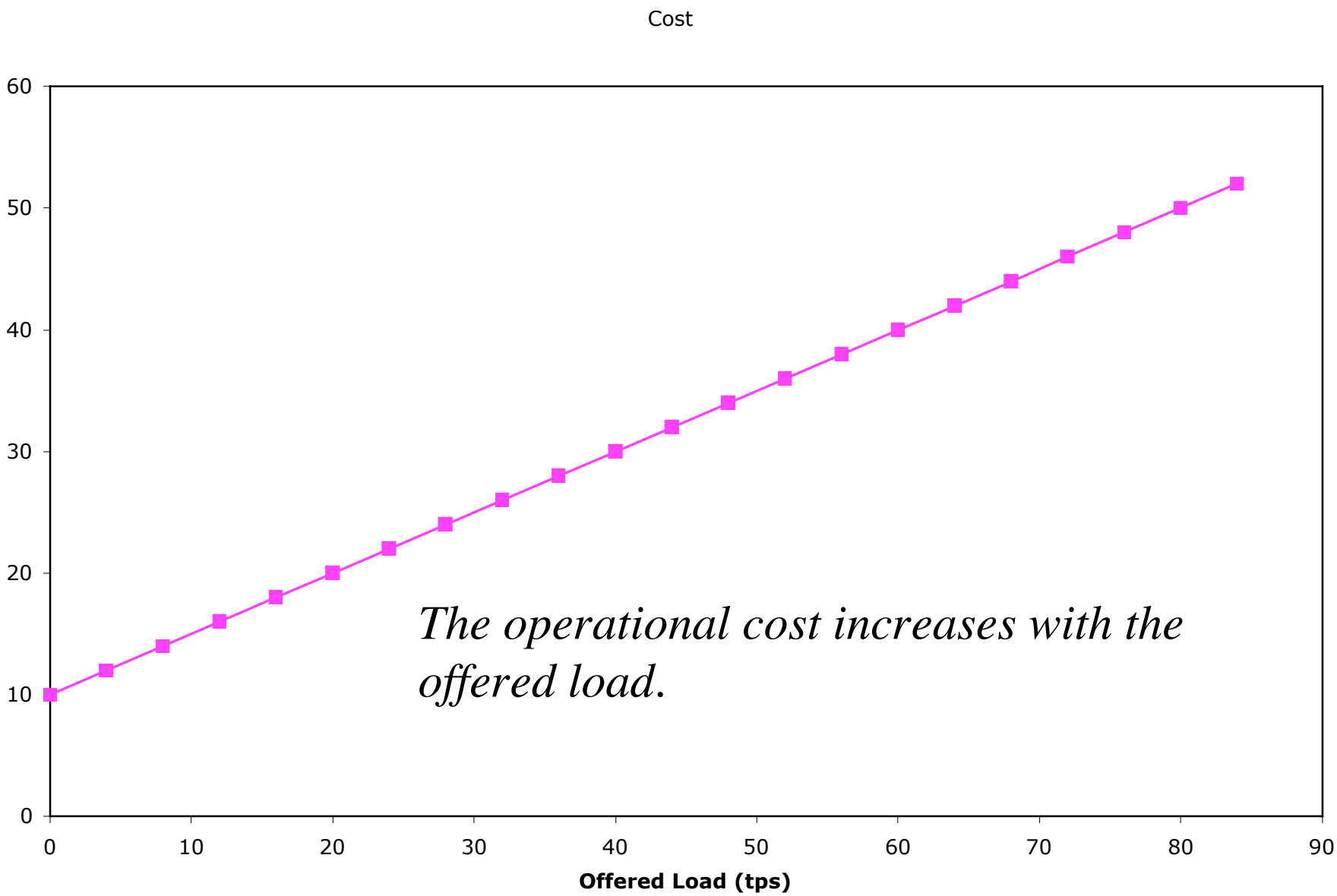
Workload:
- transactions
- HTTP requests
- video downloads
- calls to Call Center

QoS metrics:
- response time
- throughput
- availability
- page download time
- revenue throughput
- abandonment rate

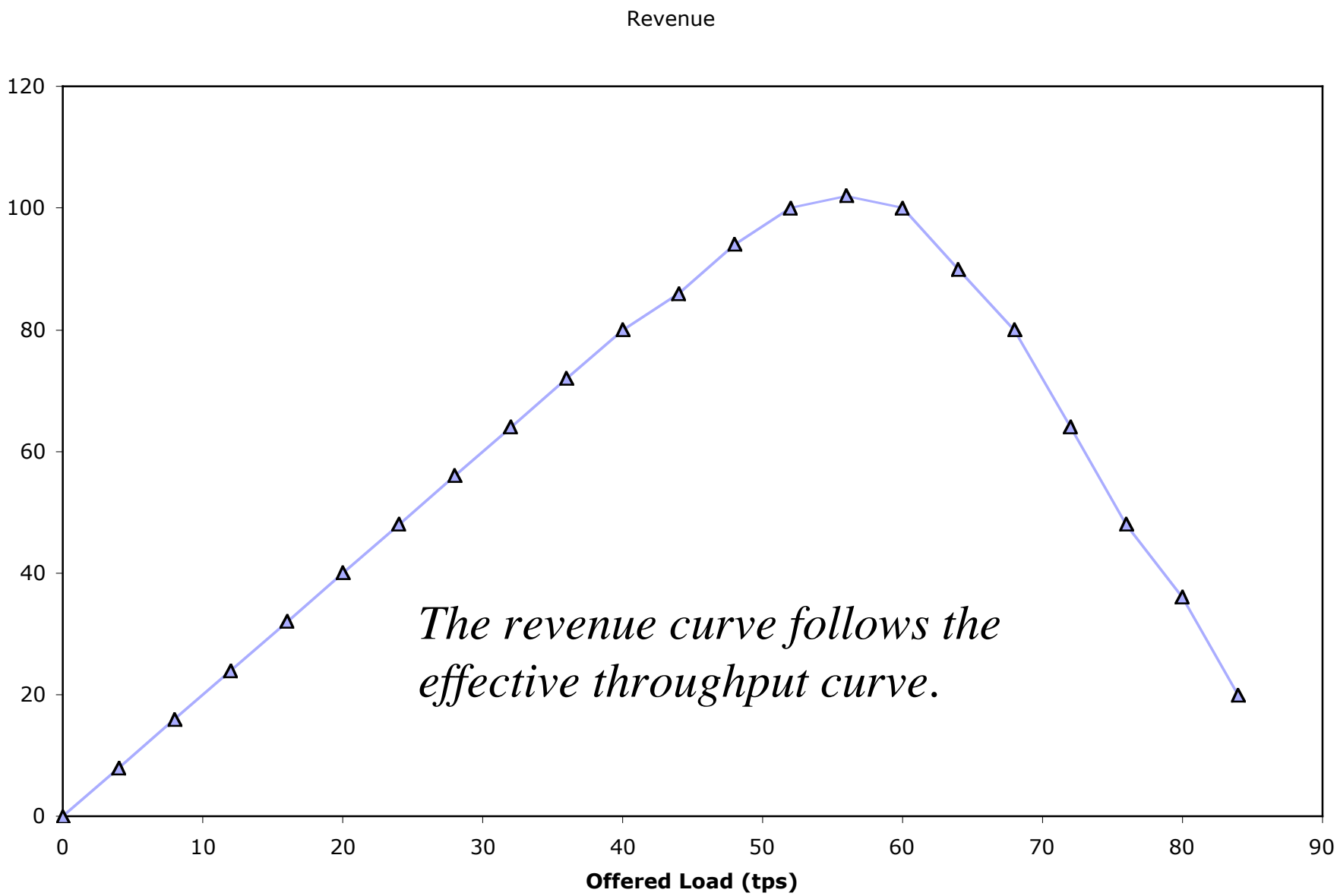
© 2005 Daniel A. Menascé. All Rights Reserved.



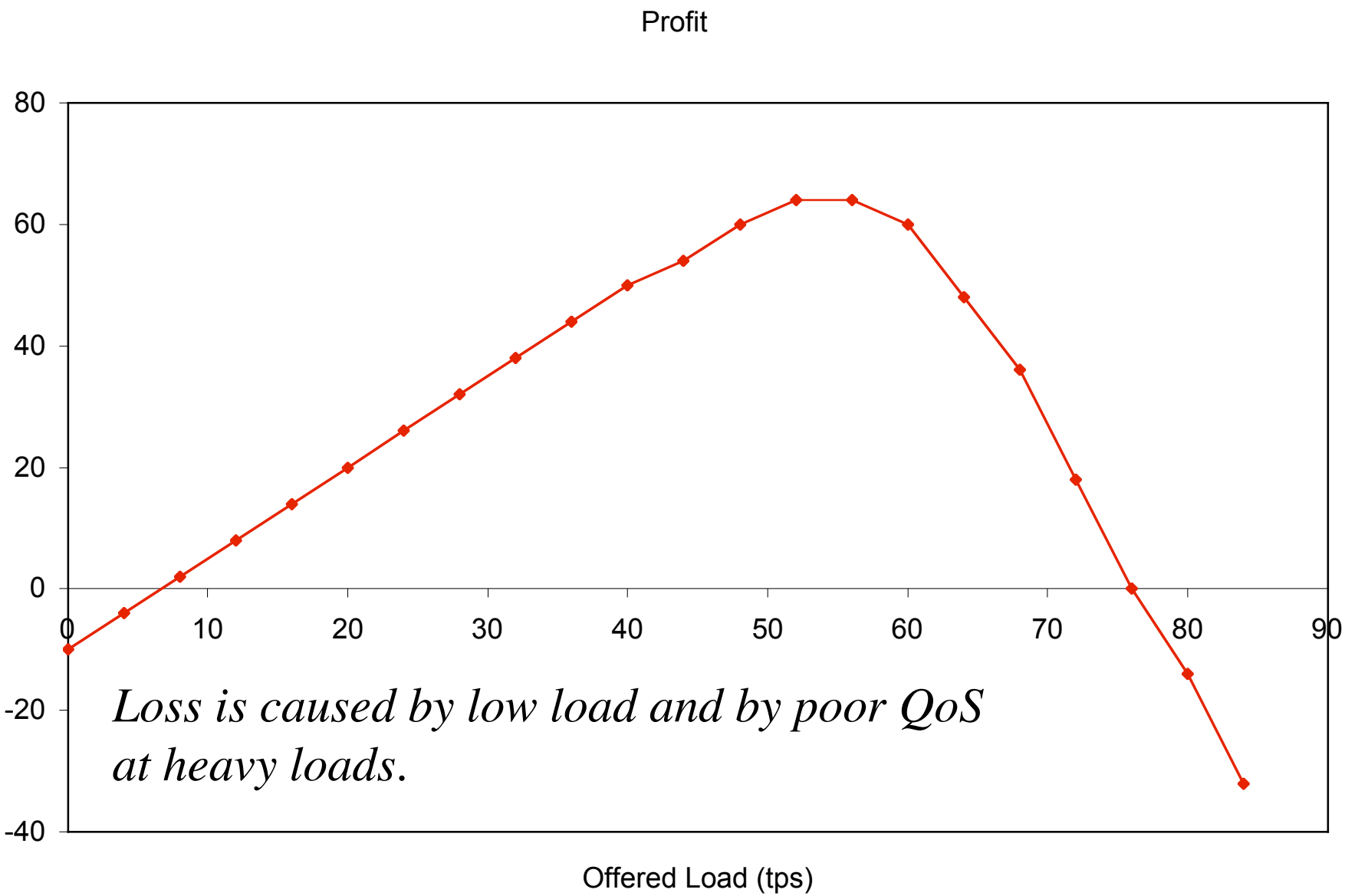
© 2005 Daniel A. Menascé. All Rights Reserved.



© 2005 Daniel A. Menascé. All Rights Reserved.



© 2005 Daniel A. Menascé. All Rights Reserved.



Loss is caused by low load and by poor QoS at heavy loads.

© 2005 Daniel A. Menascé. All Rights Reserved.

Engineering with QoS in Mind

- Poor QoS may lead to loss of human life (e.g., homeland security and disaster relief support systems) and financial losses due to customer dissatisfaction.

© 2005 Daniel A. Menascé. All Rights Reserved.

Engineering with QoS in Mind

- Poor QoS may lead to loss of human life (e.g., homeland security and disaster relief support systems) and financial losses due to customer dissatisfaction.
- QoS has to be an integral part of the design of any computer system.

© 2005 Daniel A. Menascé. All Rights Reserved.

Engineering with QoS in Mind

- Poor QoS may lead to loss of human life (e.g., homeland security and disaster relief support systems) and financial losses due to customer dissatisfaction.
- QoS has to be an integral part of the design of any computer system.
- Online computer systems (e.g., Web sites, e-commerce sites, call centers) have workloads that can be widely varying.

© 2005 Daniel A. Menascé. All Rights Reserved.

Engineering with QoS in Mind

- Poor QoS may lead to loss of human life (e.g., homeland security and disaster relief support systems) and financial losses due to customer dissatisfaction.
- QoS has to be an integral part of the design of any computer system.
- Online computer systems (e.g., Web sites, e-commerce sites, call centers) have workloads that can be widely varying.
- QoS autonomic control should be part of the design of complex online computer systems.

© 2005 Daniel A. Menascé. All Rights Reserved.

Rest of this talk…

- The design of an autonomic e-commerce system
- QoS challenges
- Concluding remarks

© 2005 Daniel A. Menascé. All Rights Reserved.

Motivation for Autonomic Systems

- Modern computer systems are complex and composed of multiple tiers

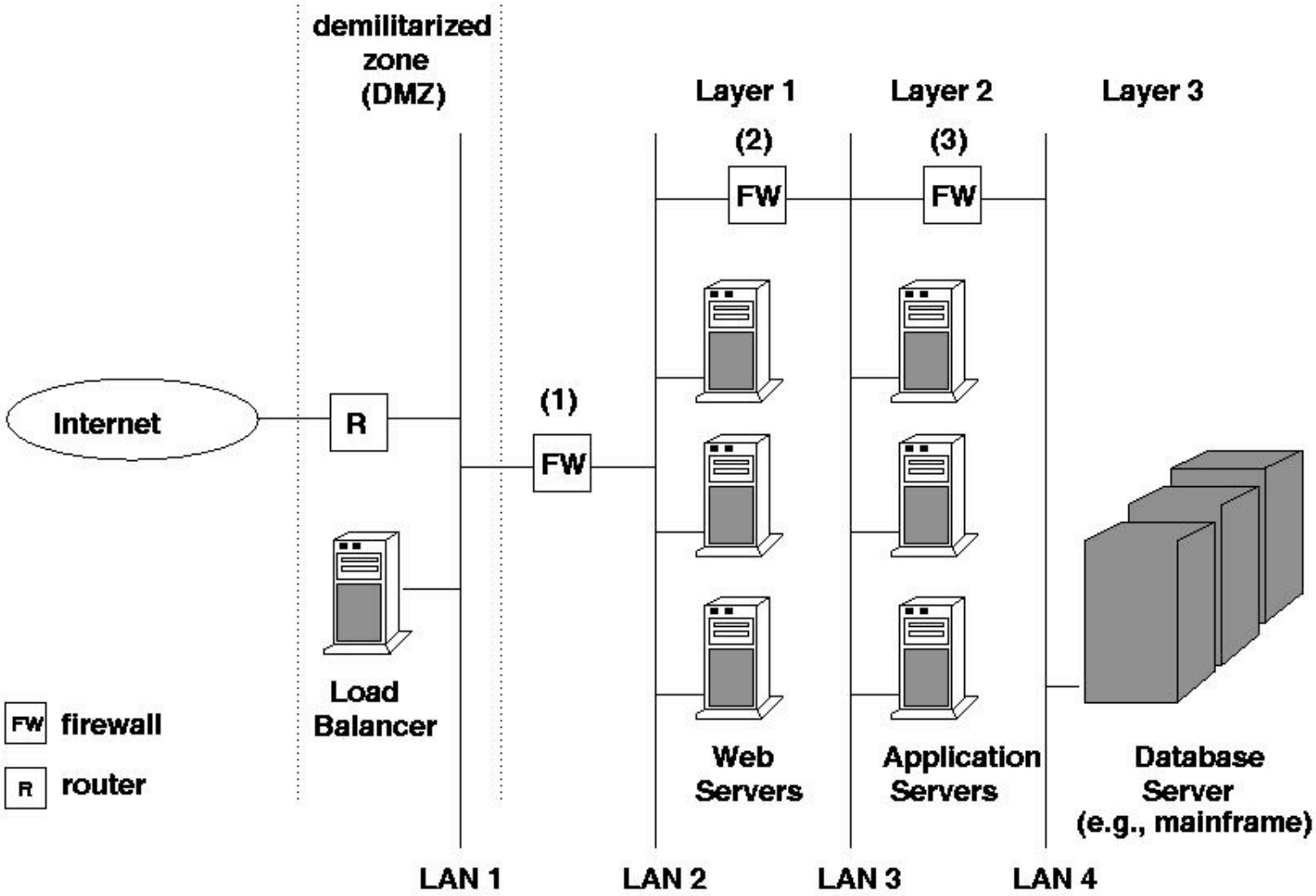
"On the Use of Online Analytic Performance Models in Self-Managing and Self-Organizing Computer Systems," (with M. Bennani and H. Ruan), in the book Self-Star Properties in Complex Information Systems, O. Babaoglu, M. Jelasity, A. Montresor, C. Fetzer , S. Leonardi, A. van Moorsel, and M. van Steen, eds., Lecture Notes in Computer Science, Vol. 3460, Springer Verlag, 2005.

"On the Use of Performance Models to Design Self-Managing Computer Systems," (with M. Bennani), Proc. 2003 Computer Measurement Group Conference, Dallas, TX, Dec. 7-12, 2003.

"Preserving QoS of E-commerce Sites Through Self-Tuning: A Performance Model Approach," (with R. Dodge and D. Barbara), Proc. 2001 ACM Conference on E-commerce, Tampa, FL, October 14-17, 2001.

© 2005 Daniel A. Menascé. All Rights Reserved.

Multi-tier Architecture



© 2005 Daniel A. Menascé. All Rights Reserved.

Motivation for Autonomic Systems

- Modern computer systems are complex and composed of multiple tiers
- The workload presents short term variations with high peak-to-average ratios

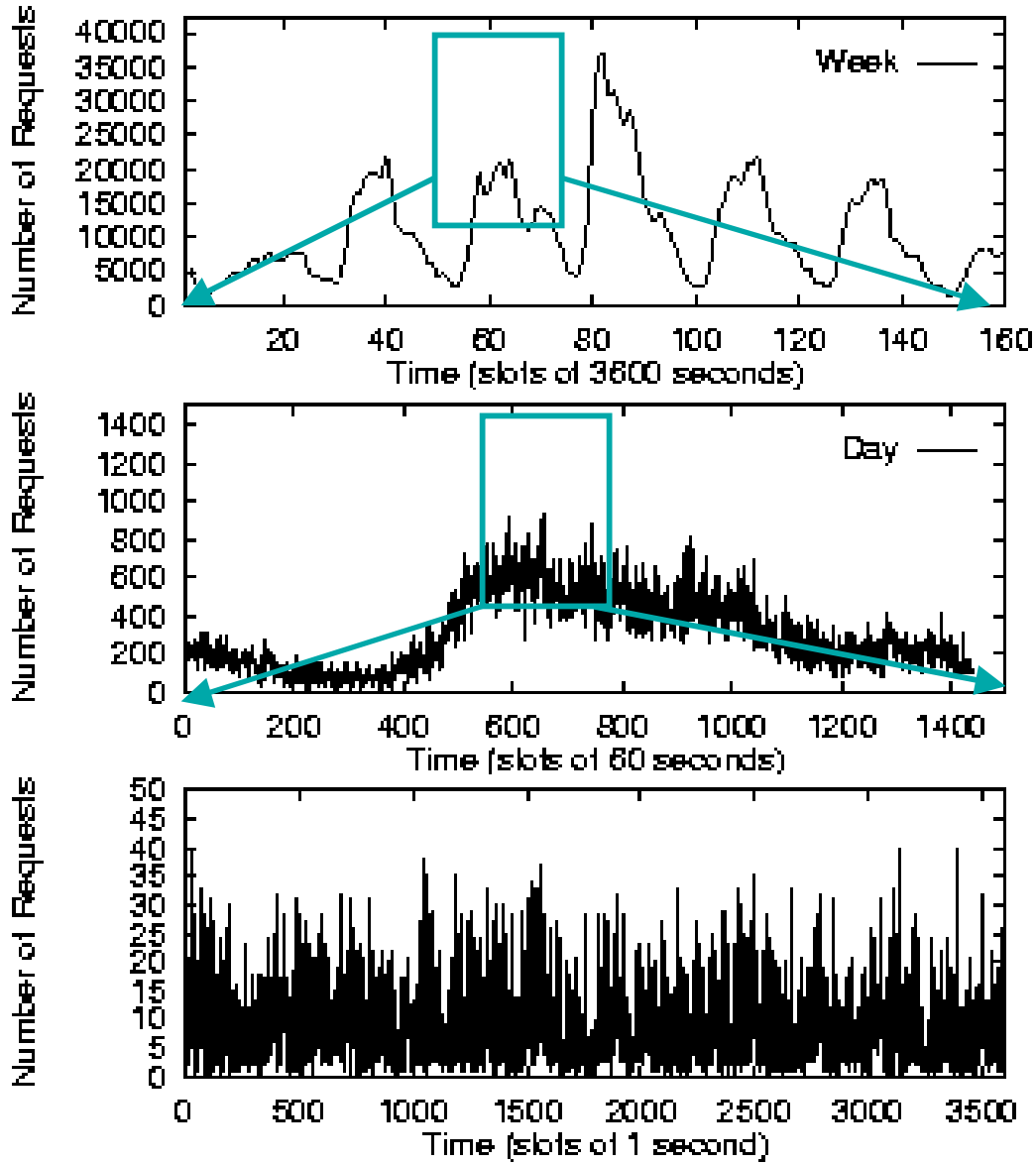
© 2005 Daniel A. Menascé. All Rights Reserved.

Multi-scale time workload variation

3600 sec

60 sec

1 sec



© 2005 Daniel A. Menascé. All Rights Reserved.

Motivation for Autonomic Systems

- Modern computer systems are complex and composed of multiple tiers
- The workload presents short term variations with high peak-to-average ratios
- Many software and hardware parameters influence system performance:
  - Manual reconfiguration is not an option
  - Need autonomic systems

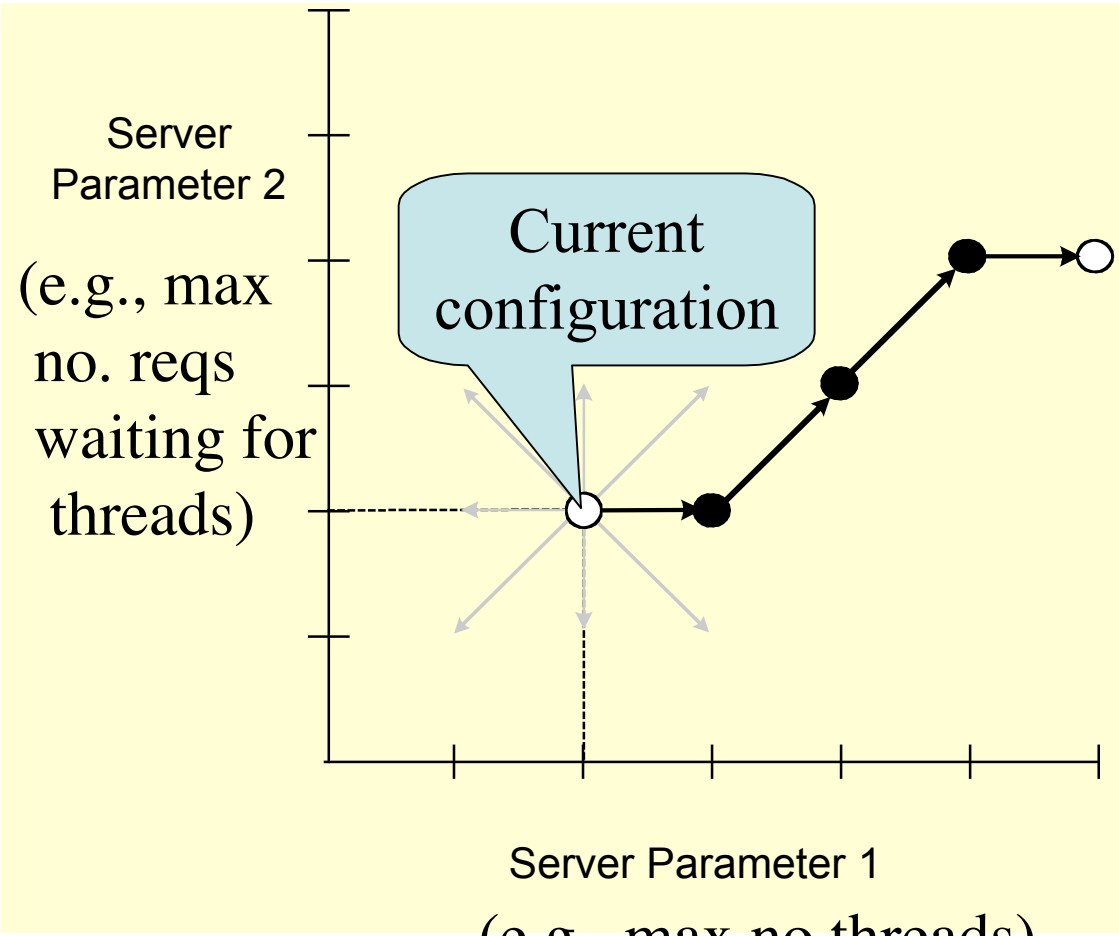
© 2005 Daniel A. Menascé. All Rights Reserved.

Examples of Configurable Parameters

Web Server (IIS 5.0)	Application Server (Tomcat 4.1)	Database Server (SQL Server 7.0)
HTTP KeepAlive	acceptCount	Cursor Threshold
Application Protection Level	minProcessors	Fill Factor
Connection Timeout	maxProcessors	Locks
Number of Connections		Max Worker Threads
Logging Location		Min Memory Per Query
Resource Indexing		Network Packet Size
Performance Tuning Level		Priority Boost
Application Optimization		Recovery Interval
MemCacheSize		Set Working Set Size
MaxCachedFileSize		Max Server Memory
ListenBacklog		Min Server Memory
MaxPoolThreads		User Connections
worker.ajp13.cachesize		

© 2005 Daniel A. Menascé. All Rights Reserved.

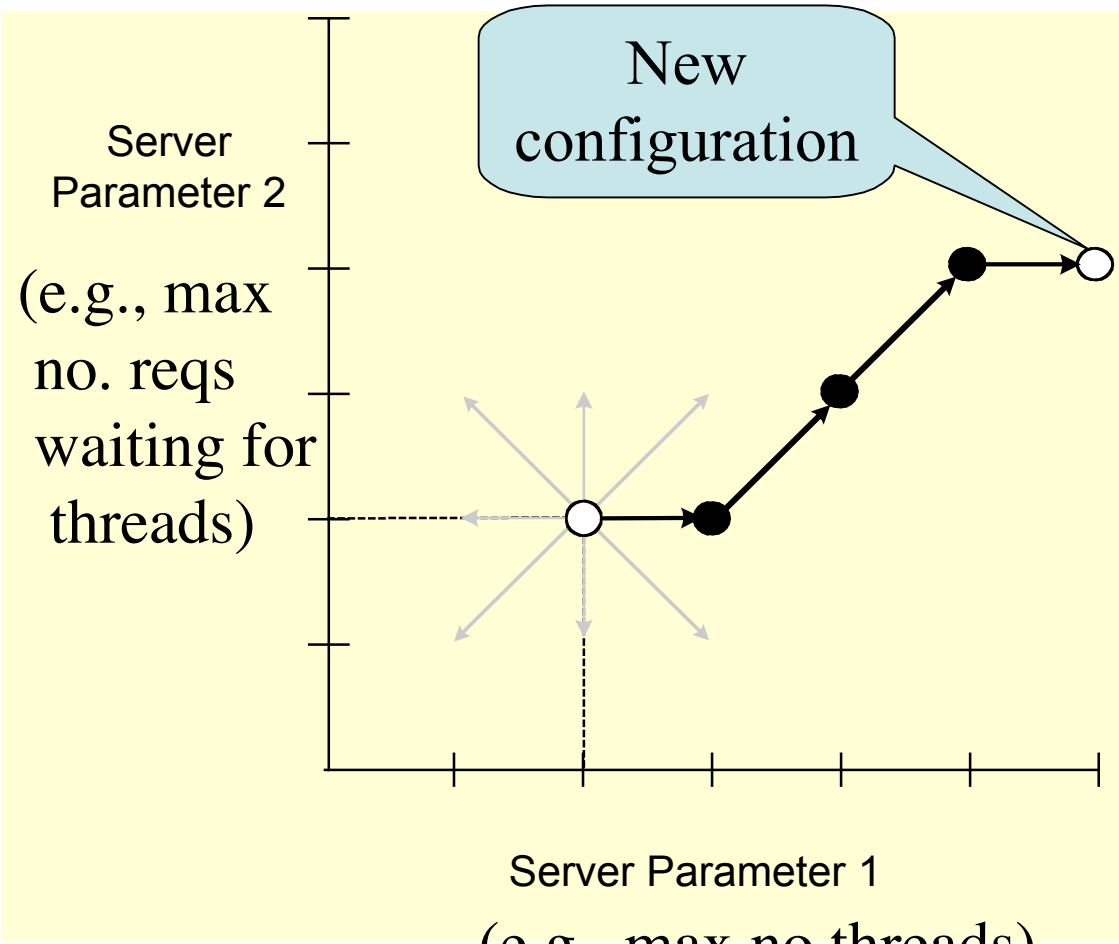
Heuristic Optimization Approach



- The space of configuration points is searched using combinatorial search techniques.
- Each point has a QoS value computed through an analytic performance model.

© 2005 Daniel A. Menascé. All Rights Reserved.

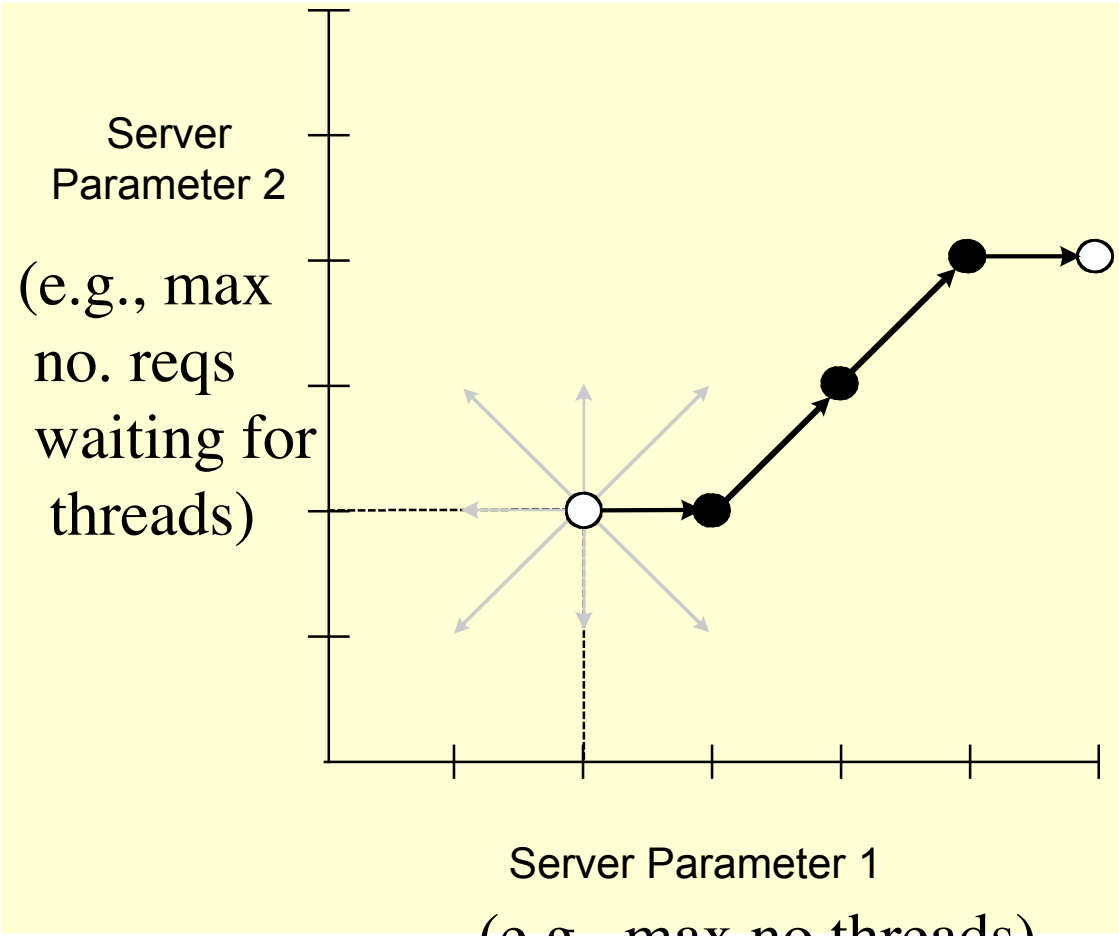
Heuristic Optimization Approach



- The space of configuration points is searched using combinatorial search techniques.
- Each point has a QoS value computed through an analytic performance model.

© 2005 Daniel A. Menascé. All Rights Reserved.

Self-Optimization Approach

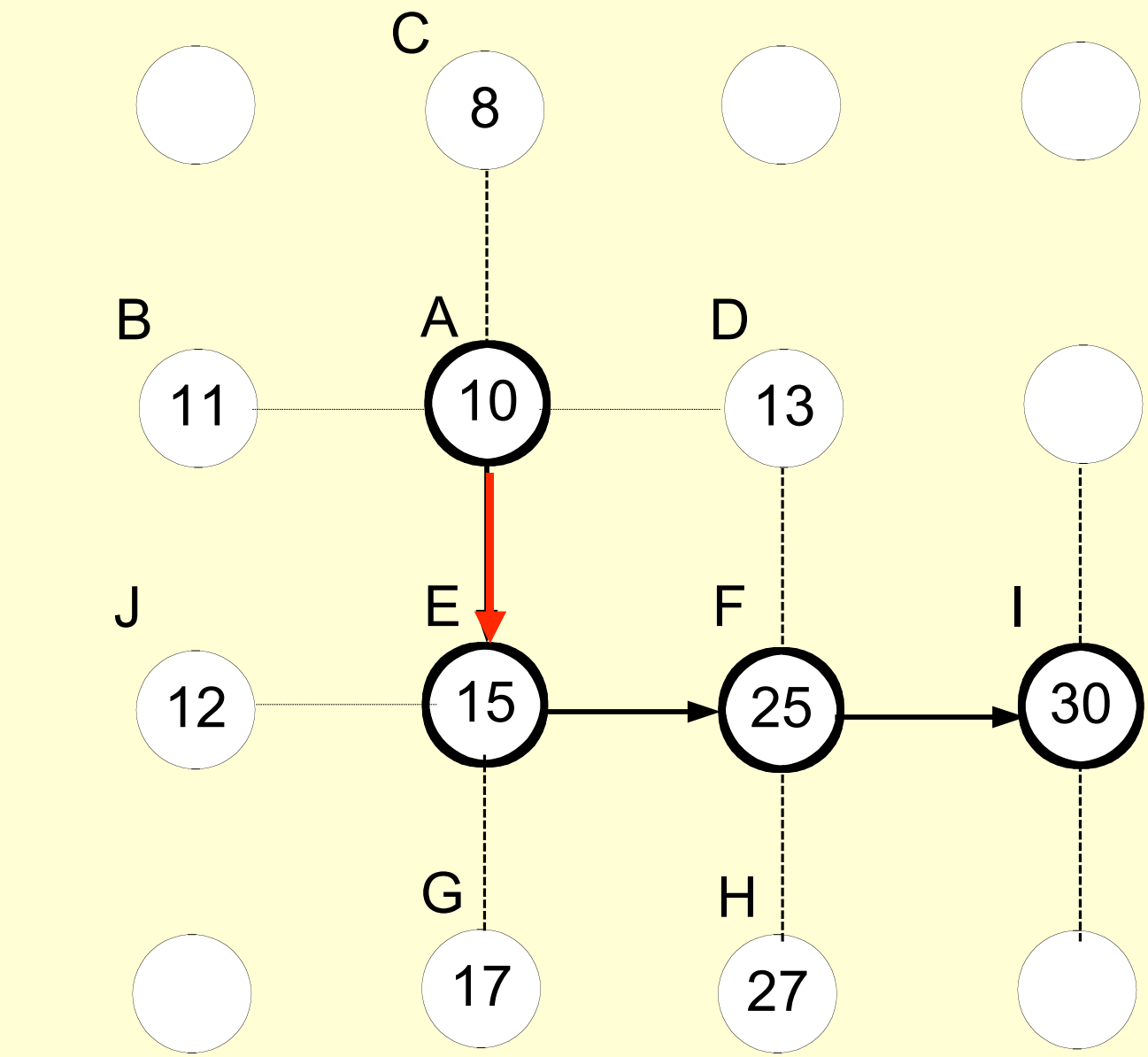


- The space of configuration points is searched using a combinatorial search technique.
- Each point has a QoS value computed through an analytic performance model.

$$QoS = f(\vec{W}, c_1, c_2, \ldots, c_m)$$

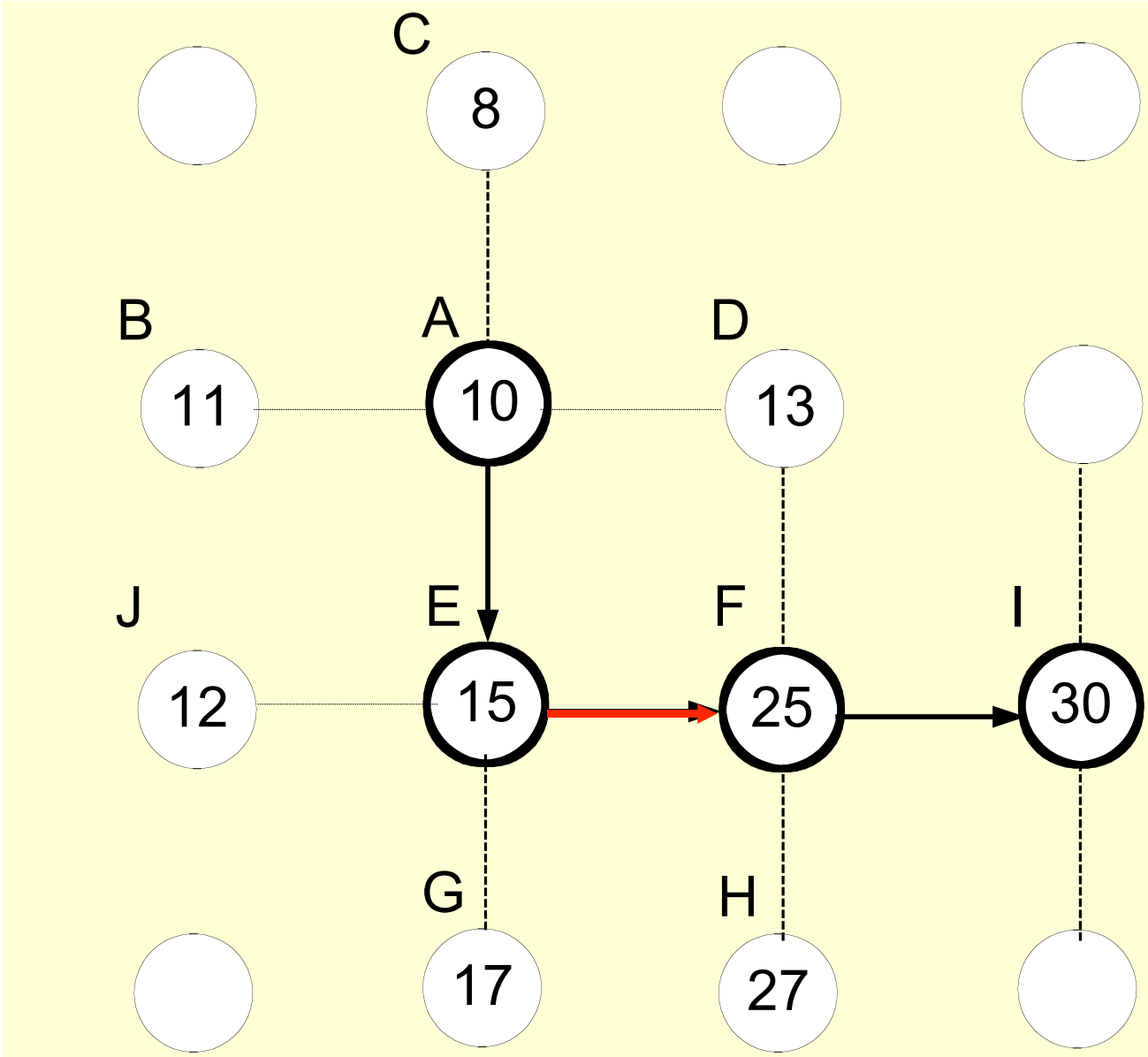
© 2005 Daniel A. Menascé. All Rights Reserved.

Hill-Climbing Search



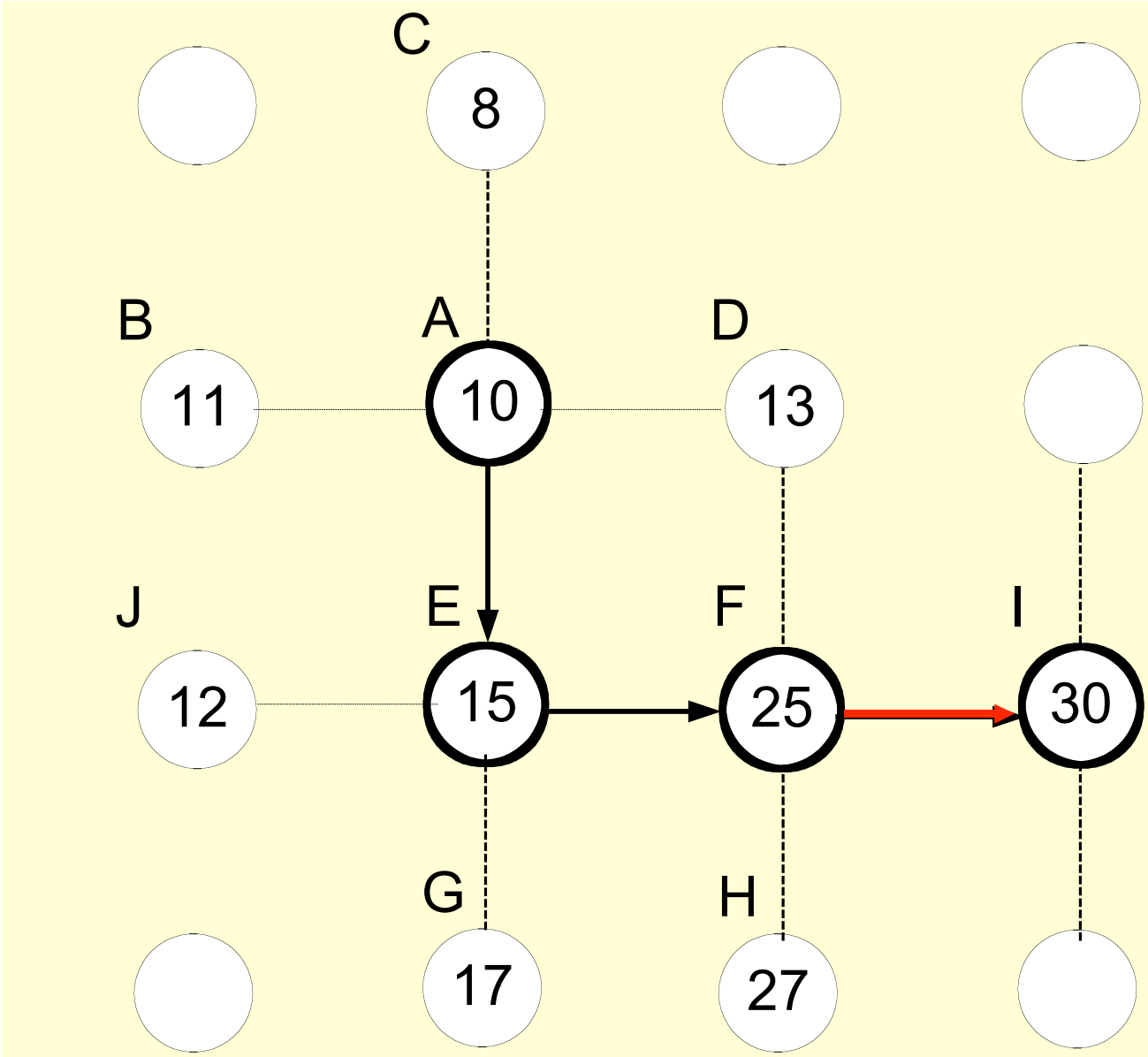
© 2005 Daniel A. Menascé. All Rights Reserved.

Hill-Climbing Search



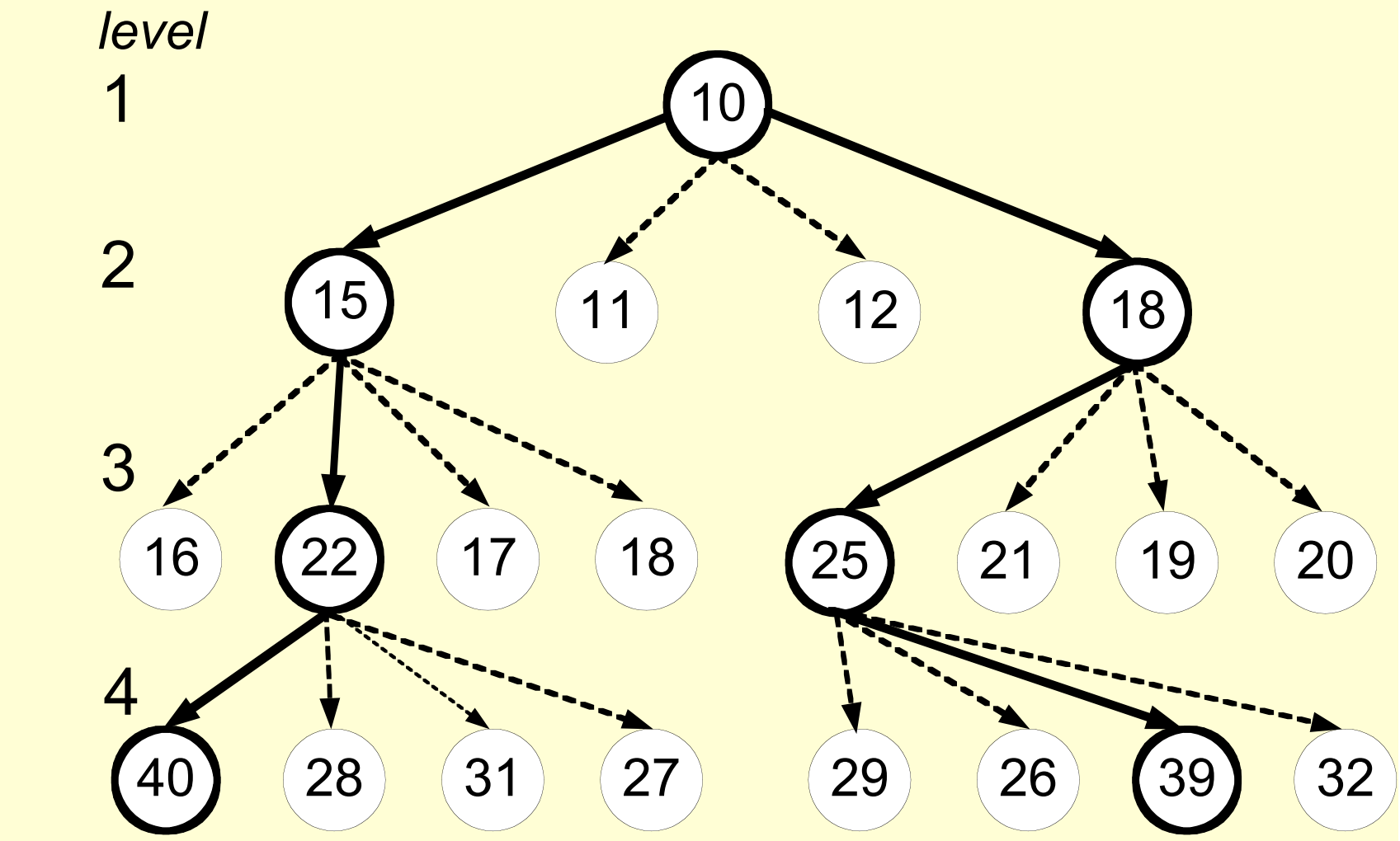
© 2005 Daniel A. Menascé. All Rights Reserved.

Hill-Climbing Search



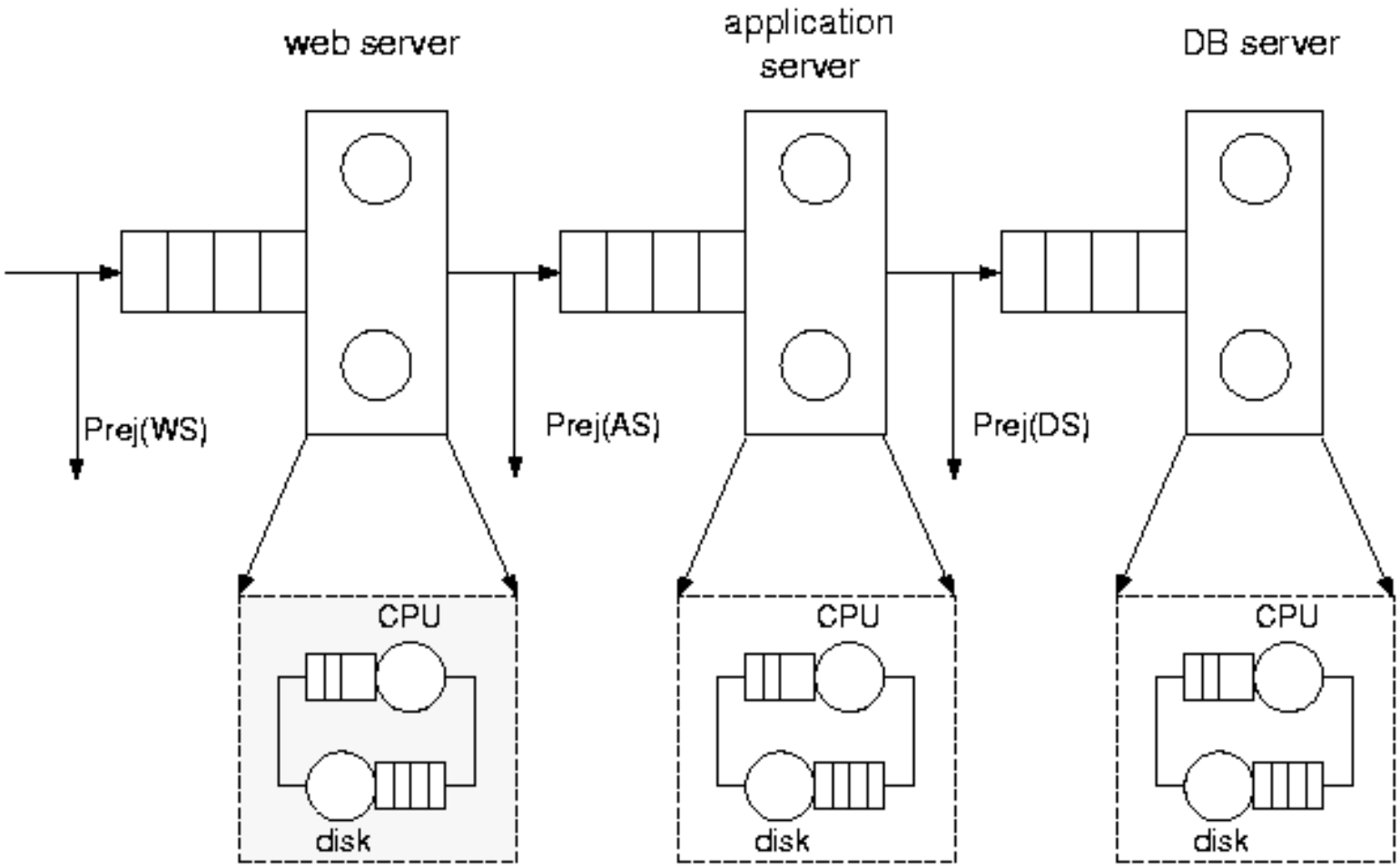
© 2005 Daniel A. Menascé. All Rights Reserved.

Beam Search



© 2005 Daniel A. Menascé. All Rights Reserved.

A Queuing Model is Used to Compute QoS Values for Each Point



© 2005 Daniel A. Menascé. All Rights Reserved.

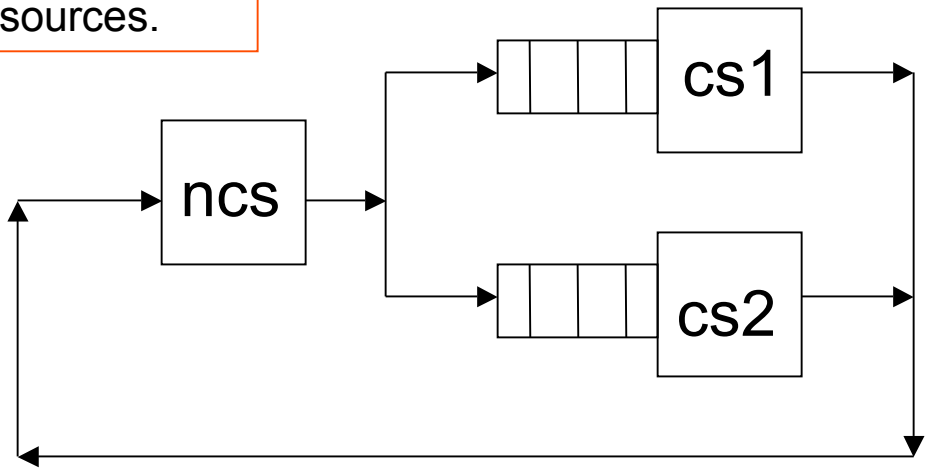
Methods for Solving Queuing Networks

- Based on Mean Value Analysis (MVA), invented by Raiser and Lavenberg and Sevcik and Mitrani in 1980.
- Have been around for over 25 years.
- Iterative techniques that converge very fast and have very good accuracy.
- Have been implemented in many commercial capacity planning software packages (e.g., Best/1 and Patrol).
- Available in the downloads of my books.

© 2005 Daniel A. Menascé. All Rights Reserved.

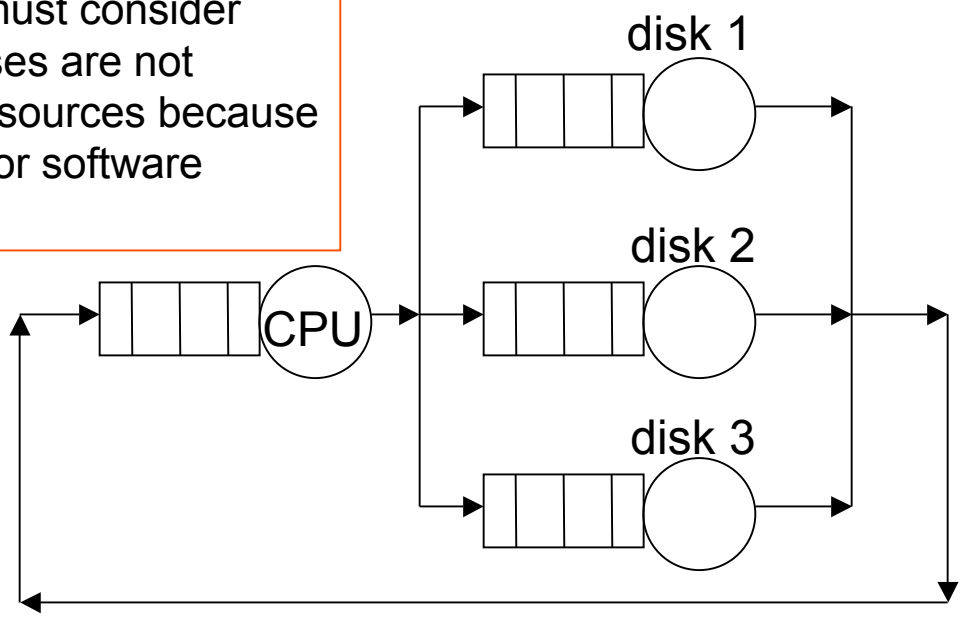
Approach

If we just consider the SQN we are ignoring time spent at the software resources due to contention for hardware resources.



Software QN

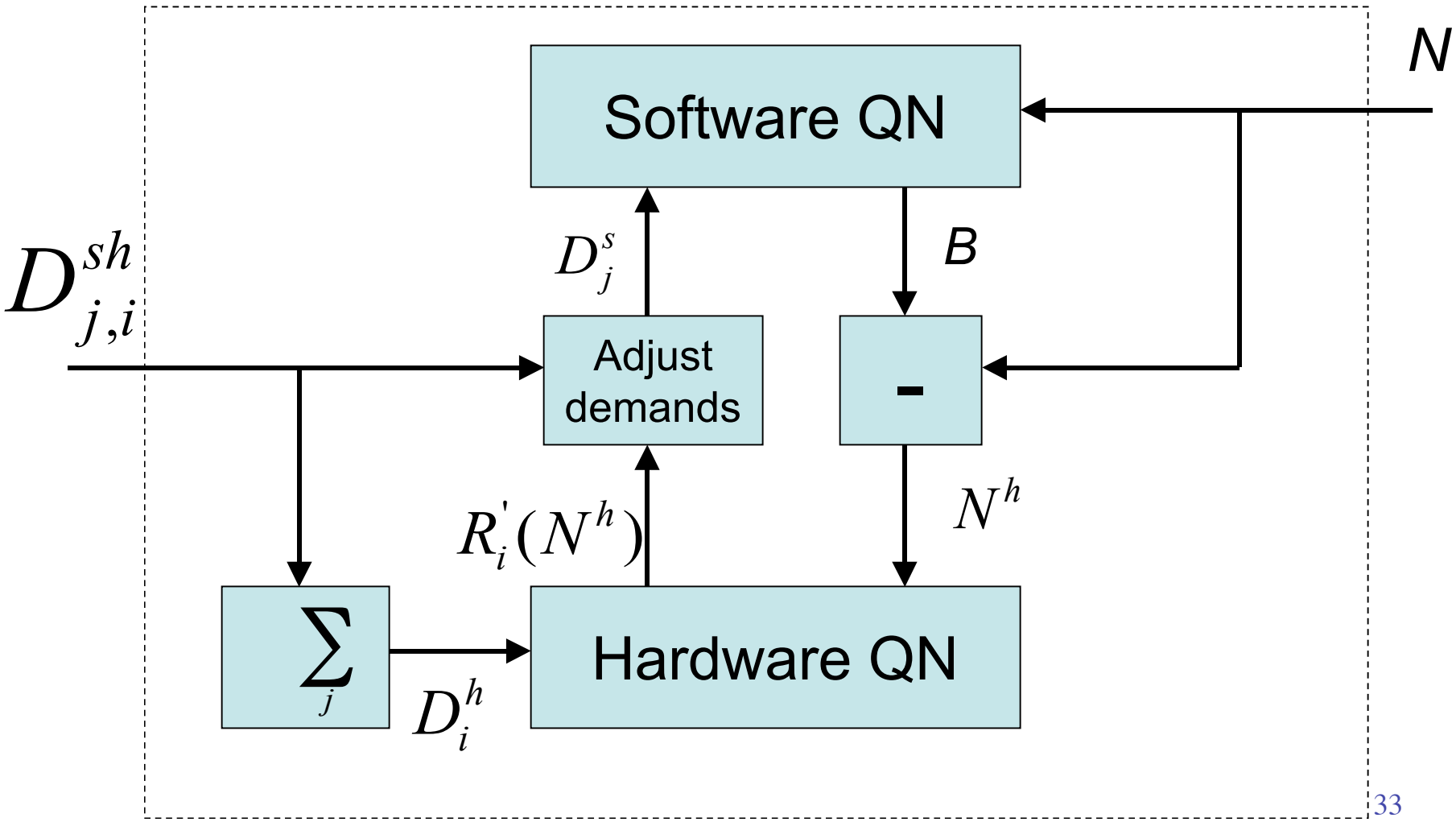
The HQN model must consider that some processes are not using hardware resources because they are blocked for software resources.



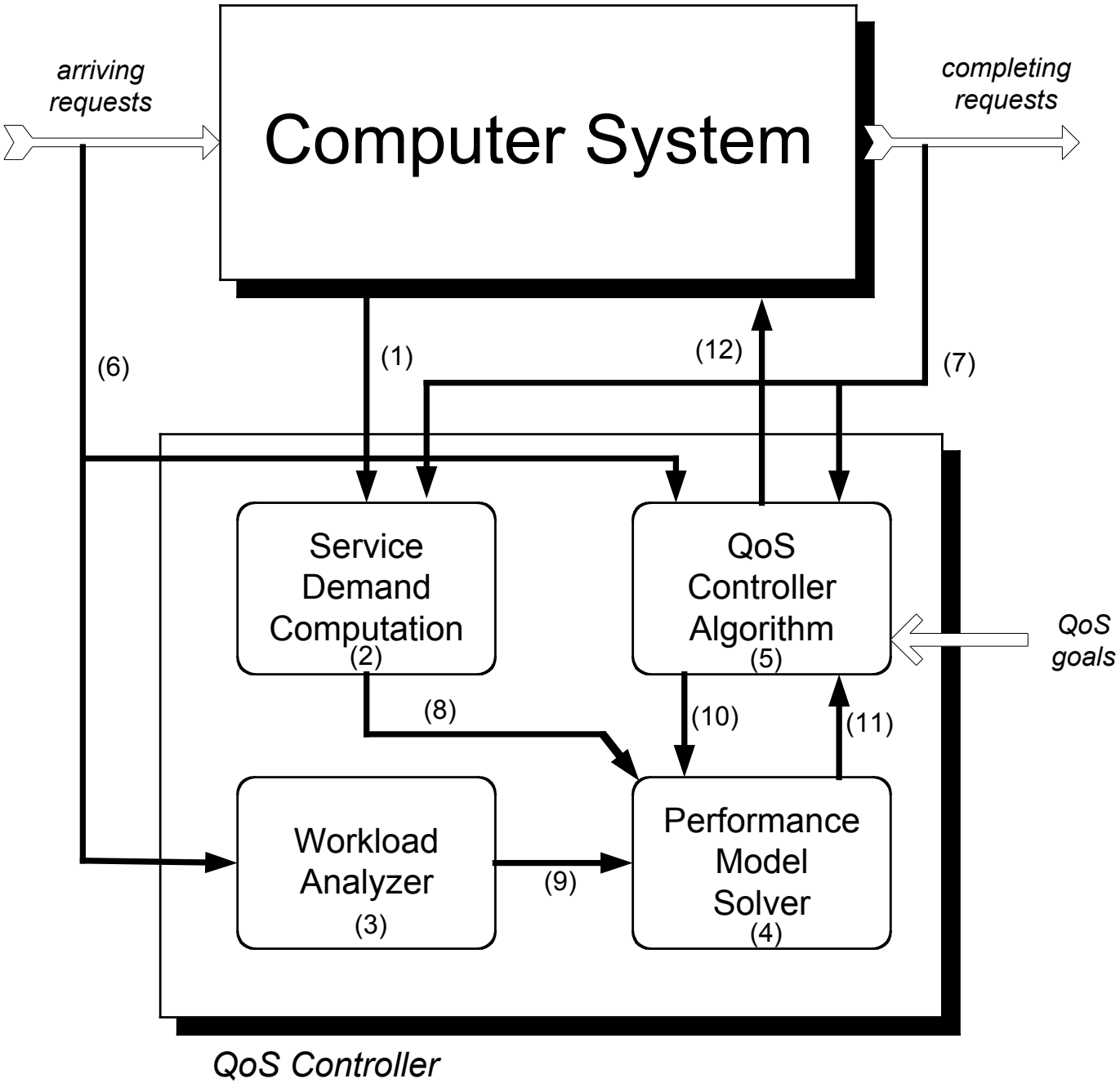
Hardware QN

© 2005 Daniel A. Menascé. All Rights Reserved.

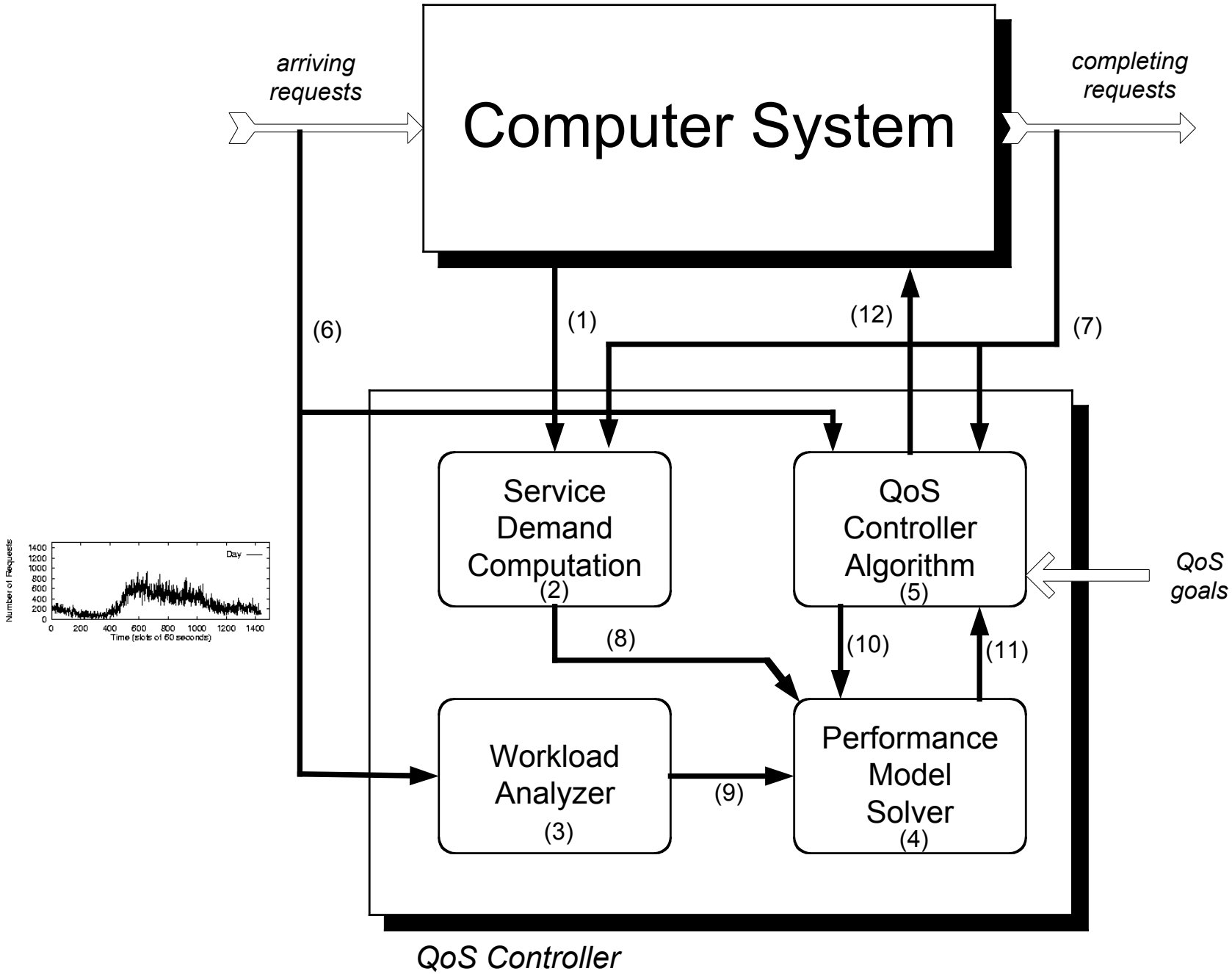
SQN-HQN Scheme



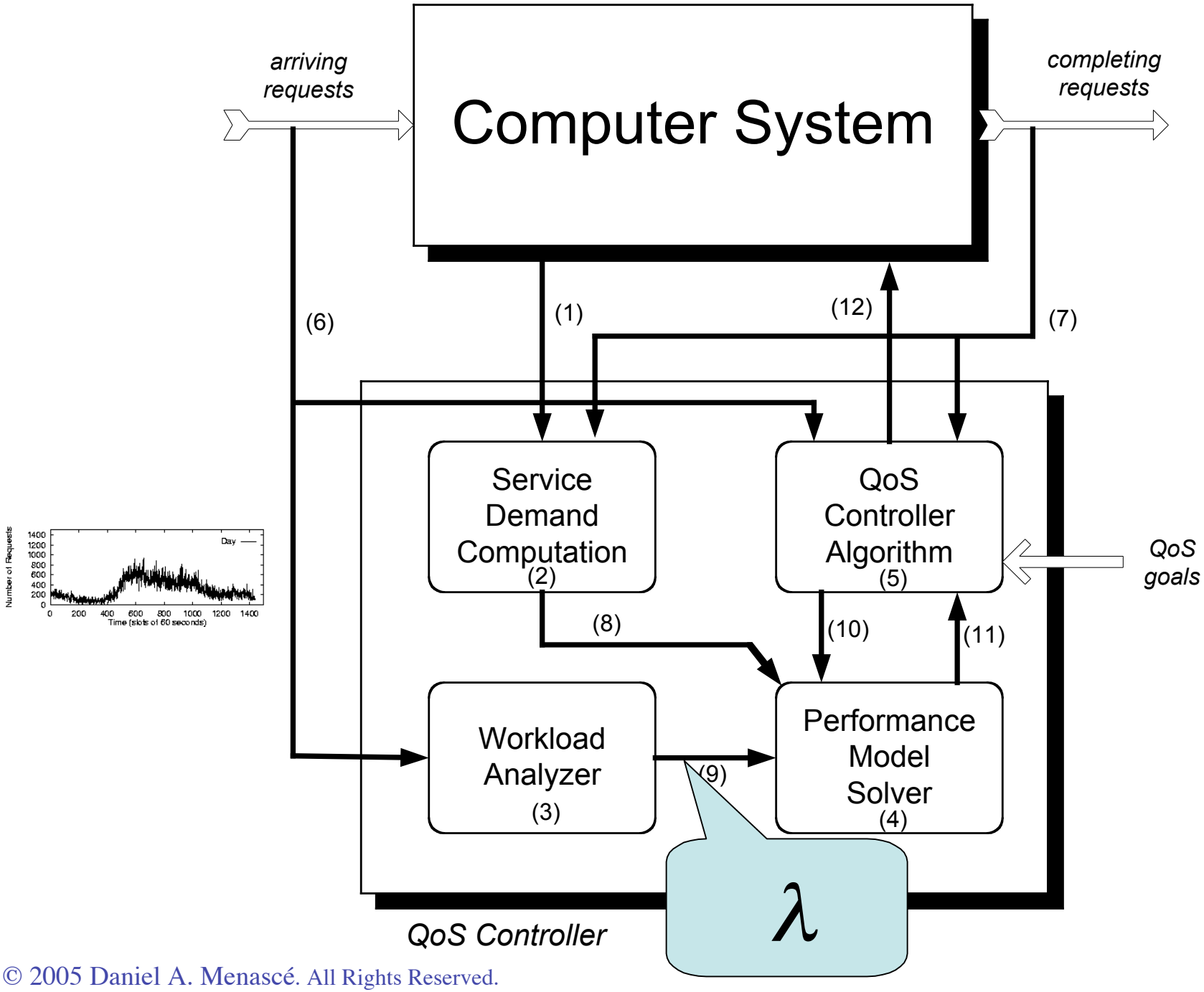
© 2005 Daniel A. Menascé. All Rights Reserved.



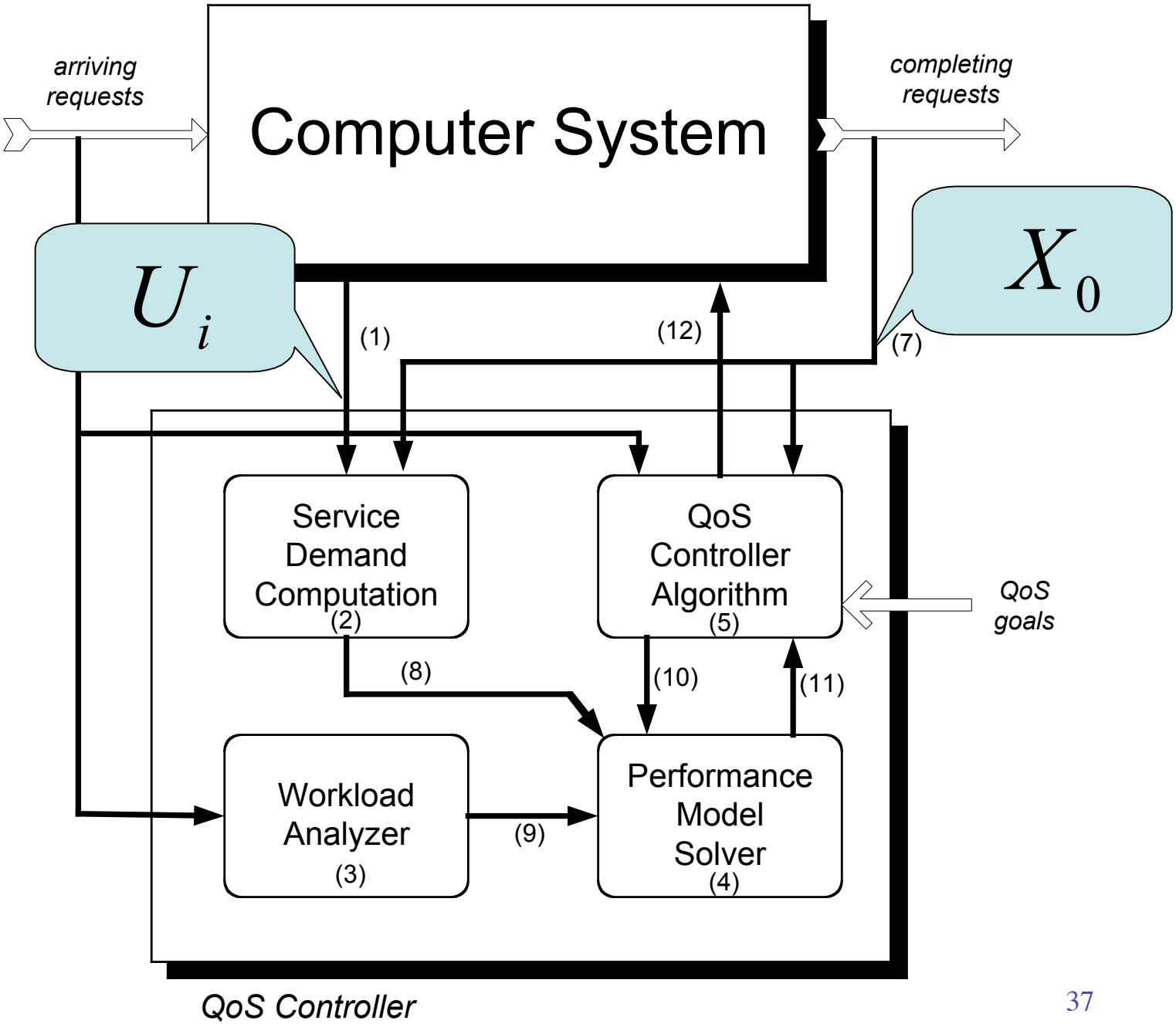
© 2005 Daniel A. Menascé. All Rights Reserved.



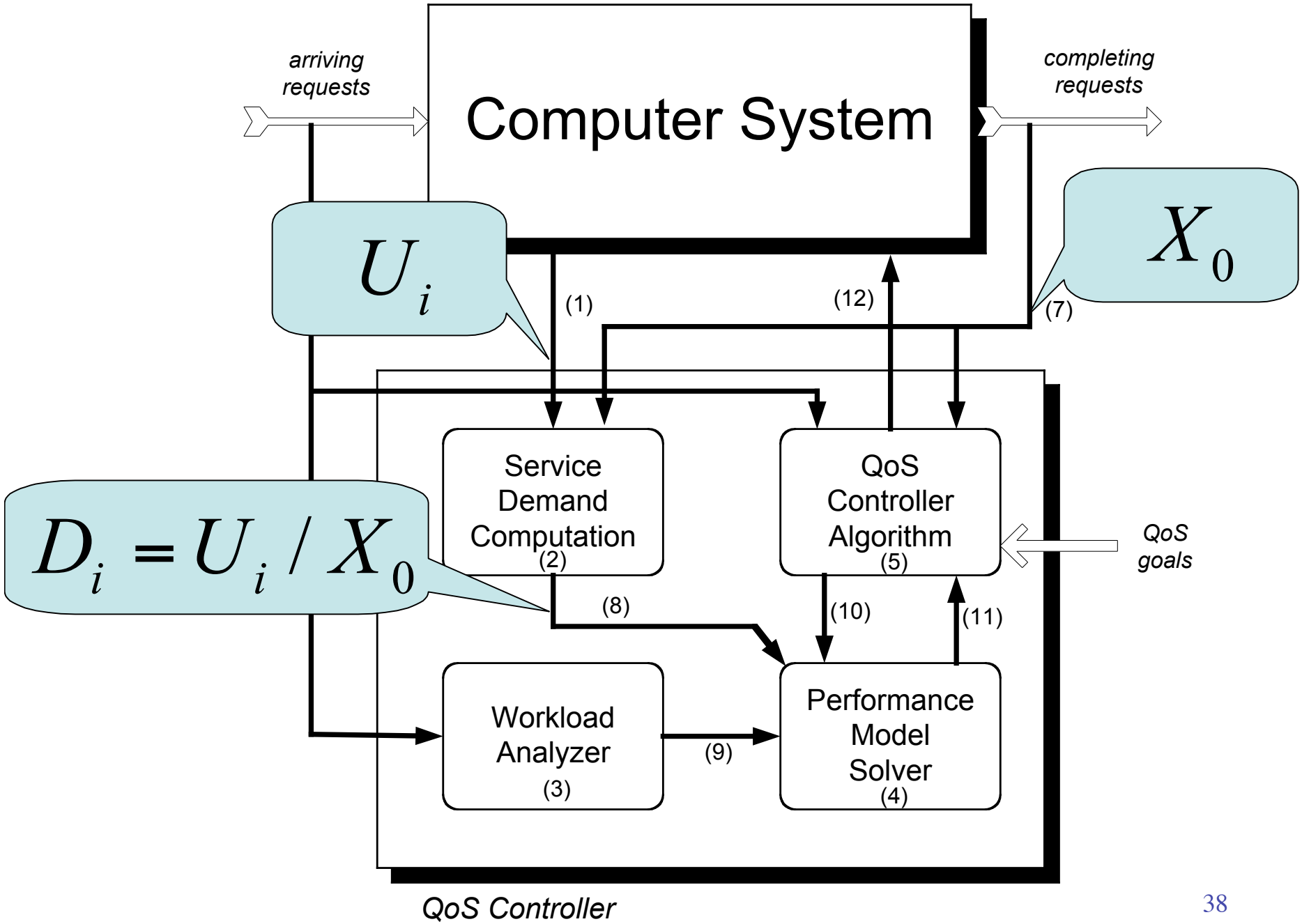
© 2005 Daniel A. Menascé. All Rights Reserved.



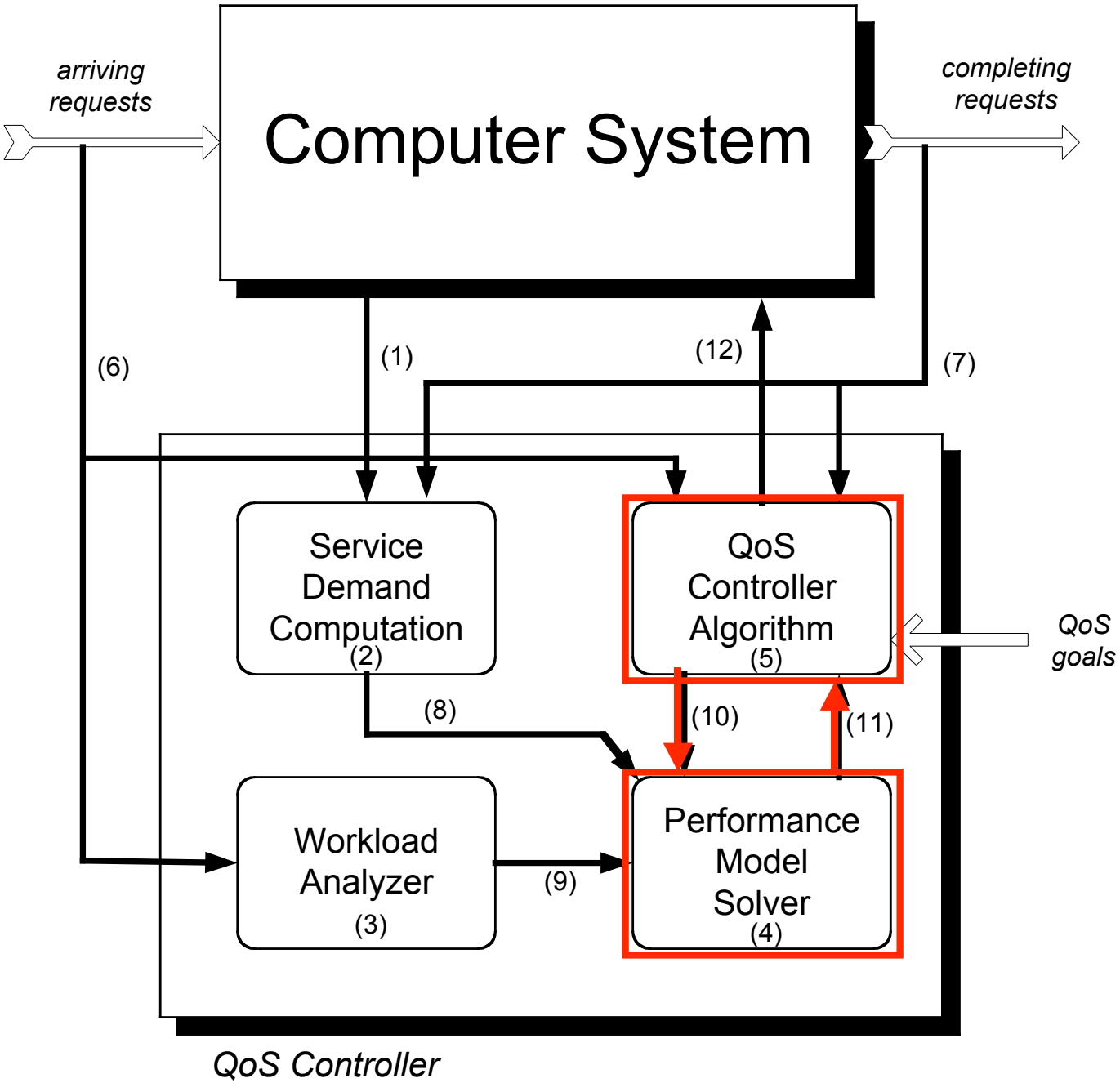
© 2005 Daniel A. Menascé. All Rights Reserved.



© 2005 Daniel A. Menascé. All Rights Reserved.

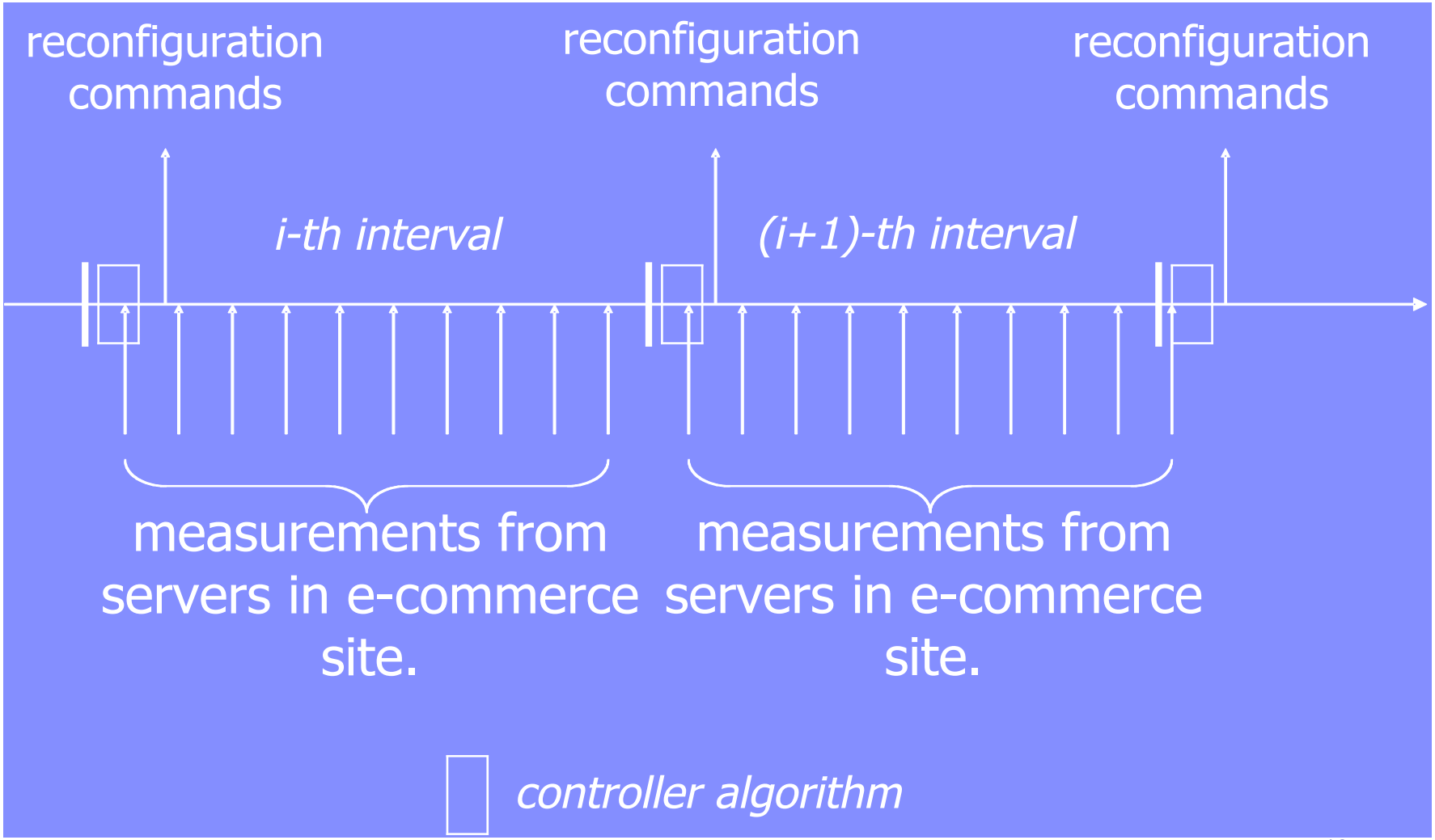


© 2005 Daniel A. Menascé. All Rights Reserved.



© 2005 Daniel A. Menascé. All Rights Reserved.

Controller Interval



© 2005 Daniel A. Menascé. All Rights Reserved.

Combined QoS Metric

$$QoS = w_R \times \Delta QoS_R + w_P \times \Delta QoS_P + w_X \times \Delta QoS_X$$

w_R, w_P, and w_X are relative weights that indicate the relative importance of response time, throughput, and probability of rejection.

© 2005 Daniel A. Menascé. All Rights Reserved.

QoS Metric

$$QoS = w_R \times \Delta QoS_R + w_P \times \Delta QoS_P + w_X \times \Delta QoS_X$$

w_R, w_P , and w_X are relative weights that indicate the relative importance of response time, throughput, and probability of rejection.

⇨ ΔQoS_R ΔQoS_P ΔQoS_X are relative deviations of the response time, throughput, and probability of rejection metrics with respect to their desired levels.

© 2005 Daniel A. Menascé. All Rights Reserved.

QoS Metric

$$QoS = w_R \times \Delta QoS_R + w_P \times \Delta QoS_P + w_X \times \Delta QoS_X$$

w_R, w_P , and w_X are relative weights that indicate the relative importance of response time, throughput, and probability of rejection.

ΔQoS_R, ΔQoS_P, and ΔQoS_X are relative deviations of the response time, throughput, and probability of rejection metrics with respect to their desired levels.

⇨ The QoS metric is a dimensionless number in the interval [-1, 1].

© 2005 Daniel A. Menascé. All Rights Reserved.

QoS Metric

$$QoS = w_R \times \Delta QoS_R + w_P \times \Delta QoS_P + w_X \times \Delta QoS_X$$

w_R, w_P, and w_X are relative weights that indicate the relative importance of response time, throughput, and probability of rejection.

ΔQoS_R, ΔQoS_P, and ΔQoS_X are relative deviations of the response time, throughput, and probability of rejection metrics with respect to their desired levels.

The QoS metric is a dimensionless number in the interval [-1, 1].

⇨ If all metrics meet or exceed their QoS targets, QoS $\geq$ 0.

© 2005 Daniel A. Menascé. All Rights Reserved.

Response Time Deviation

$$\Delta QoS_R = \frac{R_{\max} - R_{measured}}{\max(R_{\max}, R_{measured})}$$

- = 0 if the response time meets its target.
- > 0 if the response time exceeds its target.
- < 0 if the response time does not meet its target.

- $\Delta QoS_R \leq 1 - (\sum_{i=1}^{K} D_i) / R_{\max} < 1$
- $-1 < -(1 - R_{\max} / R_{measured}) \leq \Delta QoS_R$

© 2005 Daniel A. Menascé. All Rights Reserved.

Probability of Rejection Deviation

$$\Delta QoS_P = \frac{P_{\max} - P_{measured}}{\max(P_{\max}, P_{measured})}$$

- = 0 if the probability of rejection meets its target.
- > 0 and ≤ 1 if the probability of rejection exceeds its target.
- < 0 and ≥ -1 if the probability of rejection does not meet its target.

© 2005 Daniel A. Menascé. All Rights Reserved.

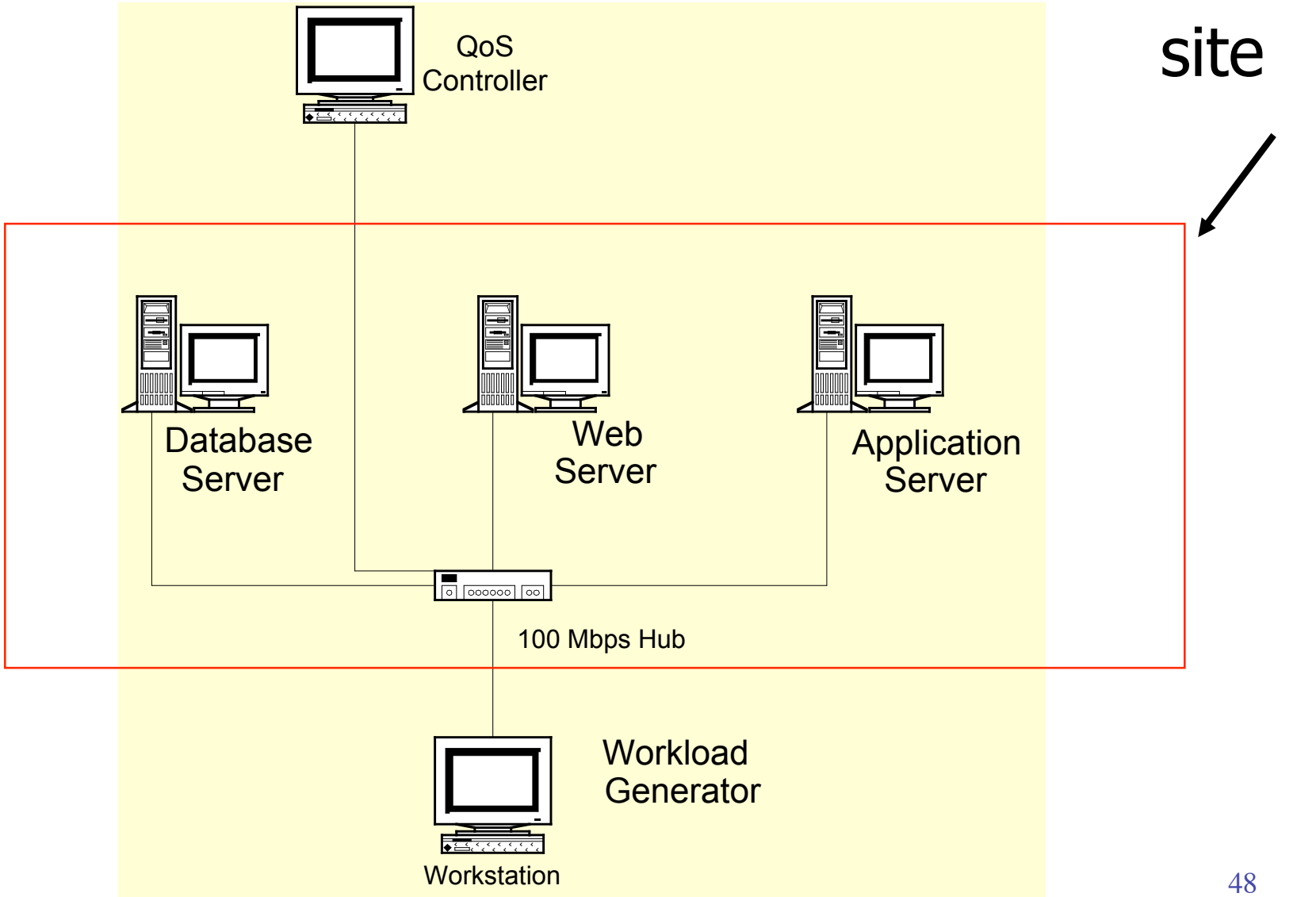
Throughput Deviation

$$\Delta QoS_X = \frac{X_{measured} - X^*_{\min}}{\max(X_{measured}, X^*_{\min})}$$

- $X^*_{\min} = \min(\lambda, X_{\min})$
- = 0 if the throughput meets its target.
- > 0 and < 1 if the throughput exceeds its target.
- < 0 and > -1 if the throughput does not meet its target.

© 2005 Daniel A. Menascé. All Rights Reserved.

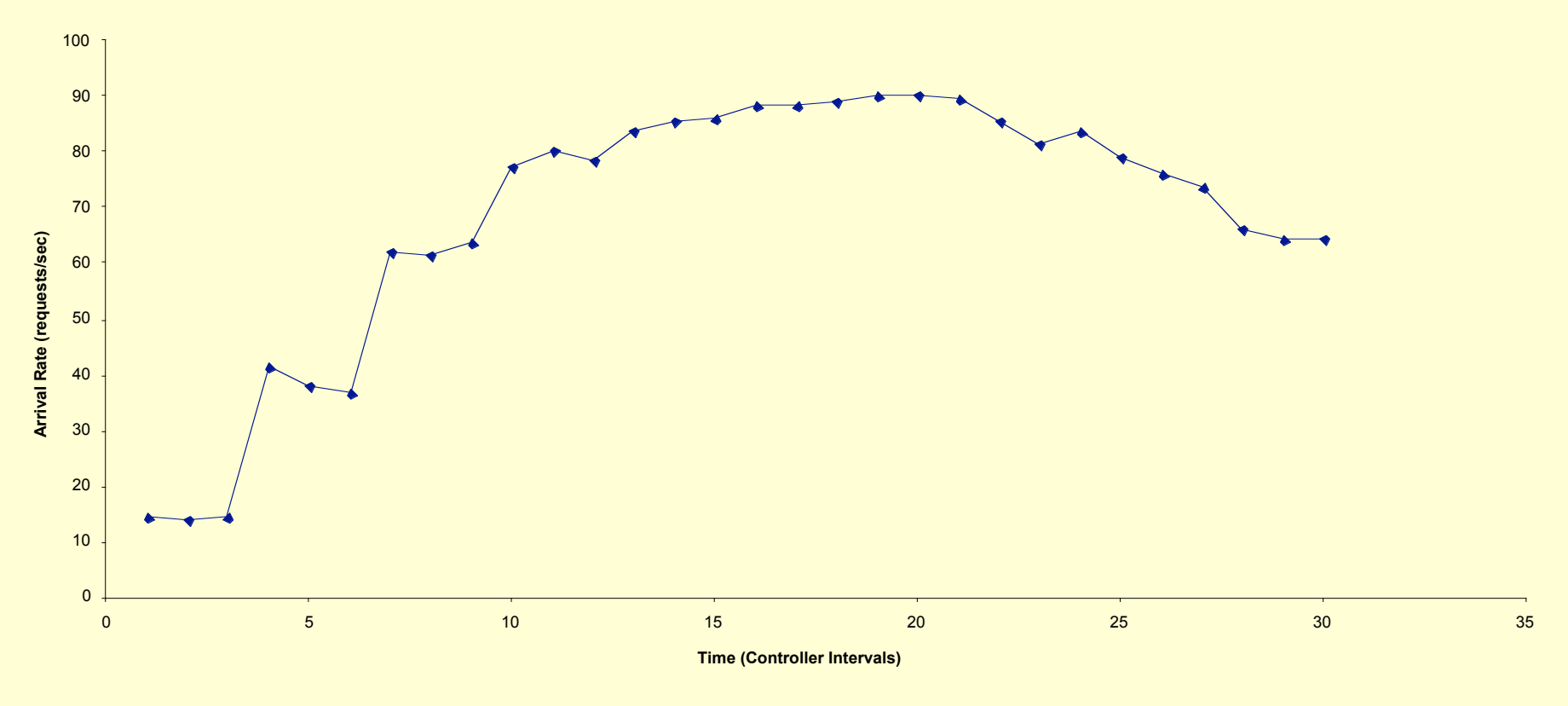
Prototype Configuration



© 2005 Daniel A. Menascé. All Rights Reserved.

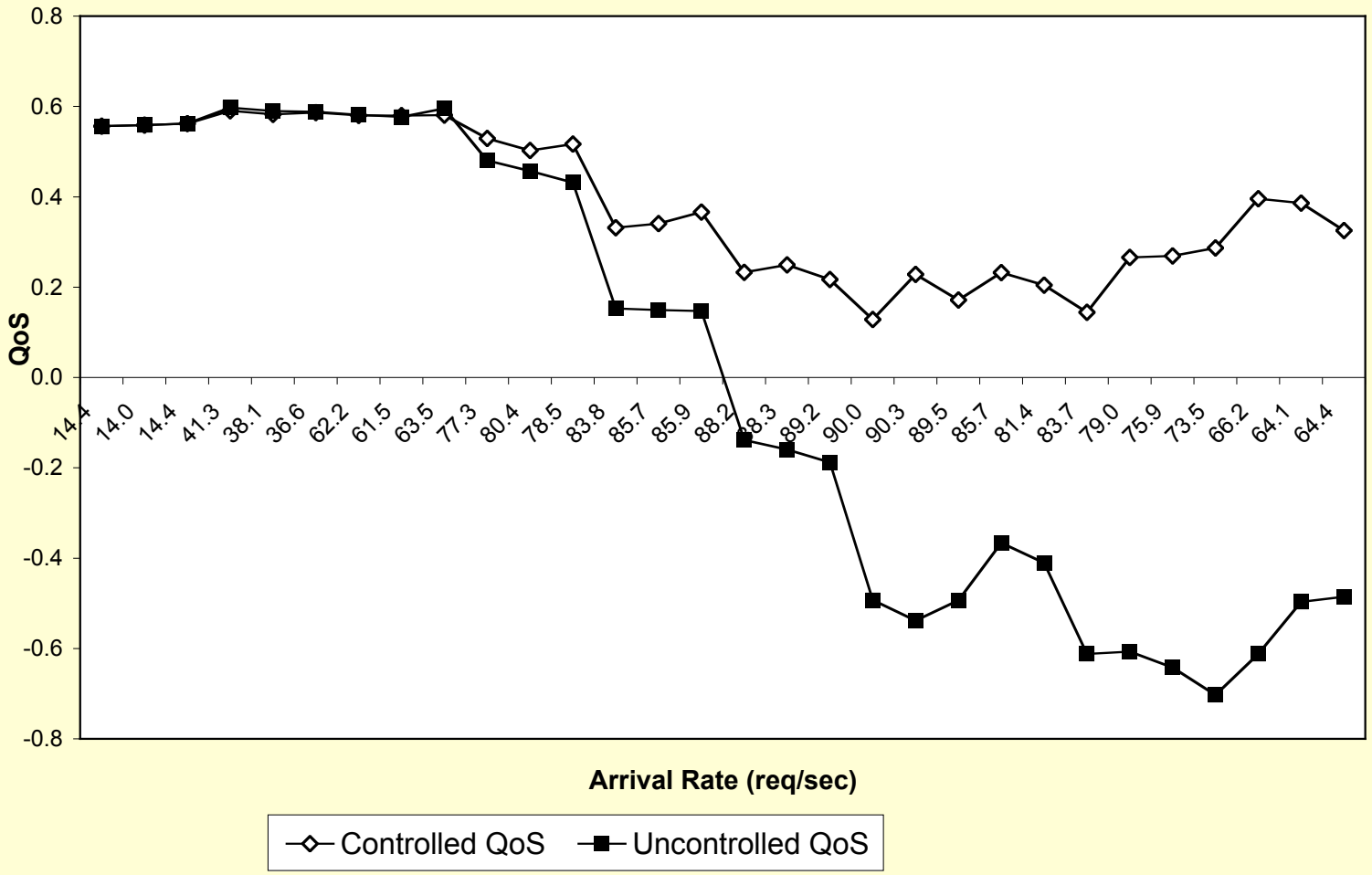
Experiment Results

Arrival rate



© 2005 Daniel A. Menascé. All Rights Reserved.

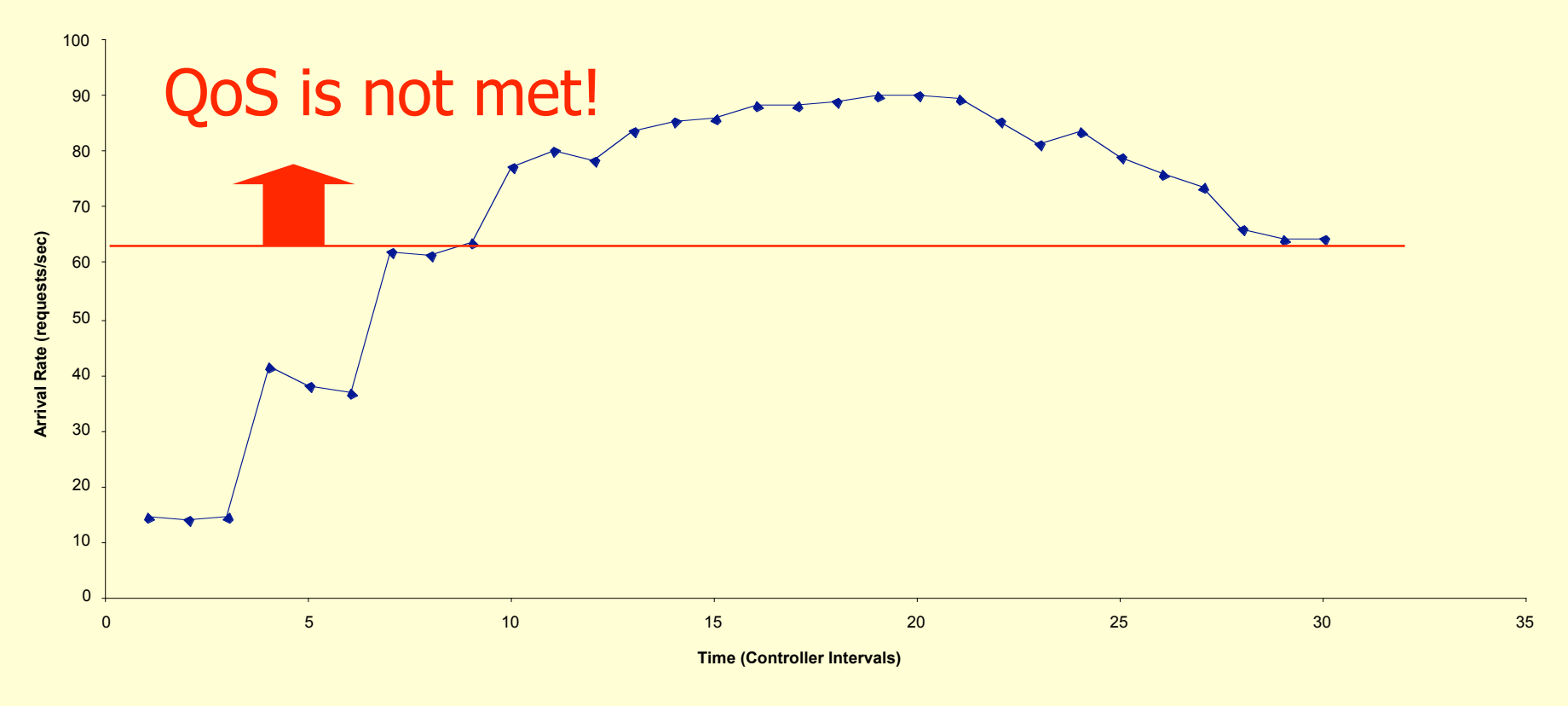
Results of QoS Controller



© 2005 Daniel A. Menascé. All Rights Reserved.

Experiment Results

Arrival rate



© 2005 Daniel A. Menascé. All Rights Reserved.

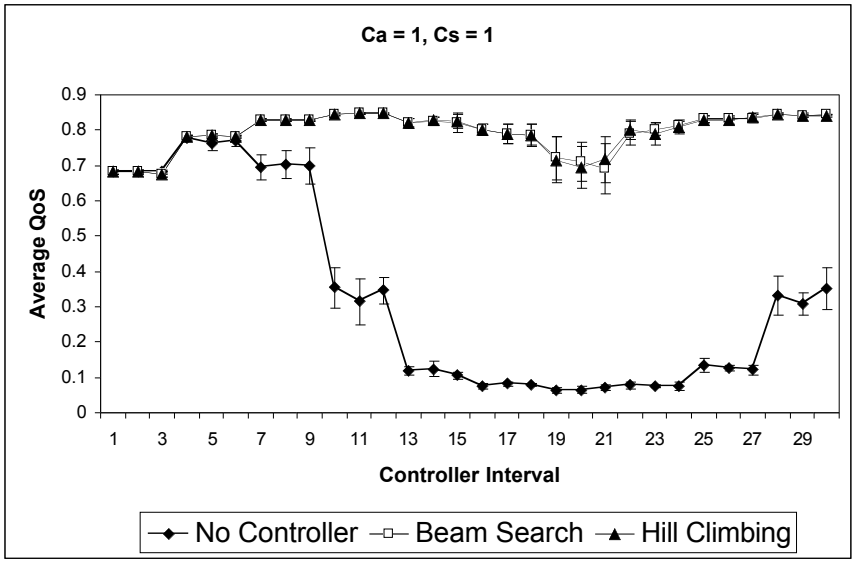
Variable inter-arrival and service times of requests

- Real workloads exhibit high variability in:
  - Traffic intensity
  - Service demands at various system resources
- Need to investigate the efficiency of the proposed self-managing technique under these conditions
- Consider variability in requests inter-arrival time and requests service times at physical resources (e.g., CPU, disk)

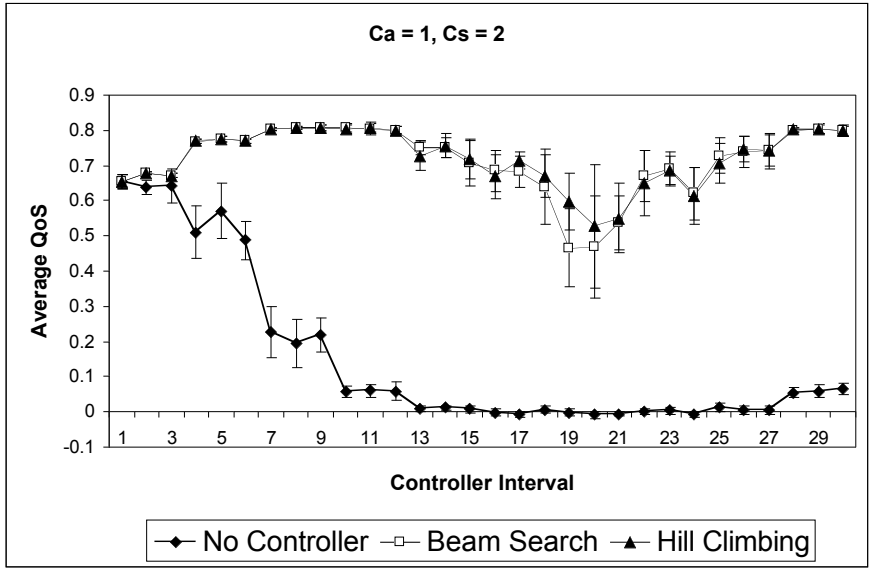
"Assessing the Robustness of Self-Managing Computer Systems under Highly Variable Workloads," (with M. Bennani), Proc. International Conf. Autonomic Computing (ICAC-04), New York, NY, May 17-18, 2004.

© 2005 Daniel A. Menascé. All Rights Reserved.

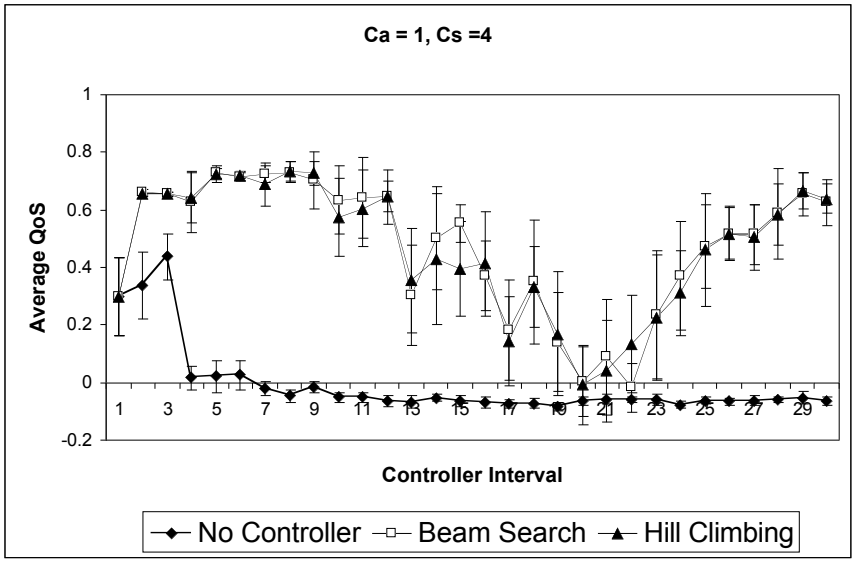
Effect of Varying the COV of the Service Time (C_a = 1)



$C_s = 1$



$C_s = 2$



$C_s = 4$

© 2005 Daniel A. Menascé. All Rights Reserved.

Observations

- Controlled system yields higher QoS values even in case of high variability in request's inter-arrival and service times
- Short term workload forecasting improves the QoS, especially when the workload intensity gets close to system saturation level
- Experiments show dynamic adjustment of CI length could improve the QoS further

© 2005 Daniel A. Menascé. All Rights Reserved.

Challenges

- Dynamically changing application structure
  - Web services and Web service composition
- Hard to characterize the workload.
  - unpredictable
  - dynamically changing services
  - application adaptation
- Difficult to build performance models.
  - moving target
- Multitude of QoS metrics at various levels of a distributed architecture
  - response time, jitter, throughput, availability, survivability, recovery time after attack/failure, call drop rate, access failure rate, packet delay, packet drop rate.
- Tradeoffs between QoS metrics (response time vs. availability, response time vs. security)

© 2005 Daniel A. Menascé. All Rights Reserved.

Challenges (cont'd)

- Need to perform transient, critical time (e.g., terrorism attack or catastrophic failures) analysis of QoS compliance. Steady-state analysis is not enough.
- Mapping of global SLAs to local SLAs
  - Cost and pricing issues.
- QoS monitoring, negotiation, and enforcement.
- Platform-neutral representation of QoS goals and contracts.
- Resource management: resource reservation, resource allocation, admission control.
  - non-dedicated resources

© 2005 Daniel A. Menascé. All Rights Reserved.

What we need …

- Design self-regulating (autonomic) systems
- Embed within each system component:
  - Monitoring capabilities
  - Negotiation capabilities (requires predictive modeling power)
  - Self-protection and recovery capabilities (for attacks, failures, and overloads)
- Push more of the desired system features to individual components
- Design QoS-aware middleware
  - QoS negotiation protocols
  - Mapping of global to local SLAs
  - QoS monitoring

© 2005 Daniel A. Menascé. All Rights Reserved.

Concluding Remarks

- QoS is an important component in the design of complex networked computer systems.
- The risks and costs of overlooking QoS can be very high.
- Performance models can be used to predict the performance of computer systems as well as to design QoS controllers.

© 2005 Daniel A. Menascé. All Rights Reserved.

Thanks!

Questions?

© 2005 Daniel A. Menascé. All Rights Reserved.