# Self-Managed Computer Systems: Foundations and Examples

Danny Menascé, University Professor
Department of Computer Science
George Mason University
www.cs.gmu.edu/faculty/menasce.html

# COMPLEXITY OF COMPUTER SYSTEMS

Amazon EC2

salesforce

facebook

Google

YouTube

LinkedIn

YAHOO!

amazon Prime

travelocity

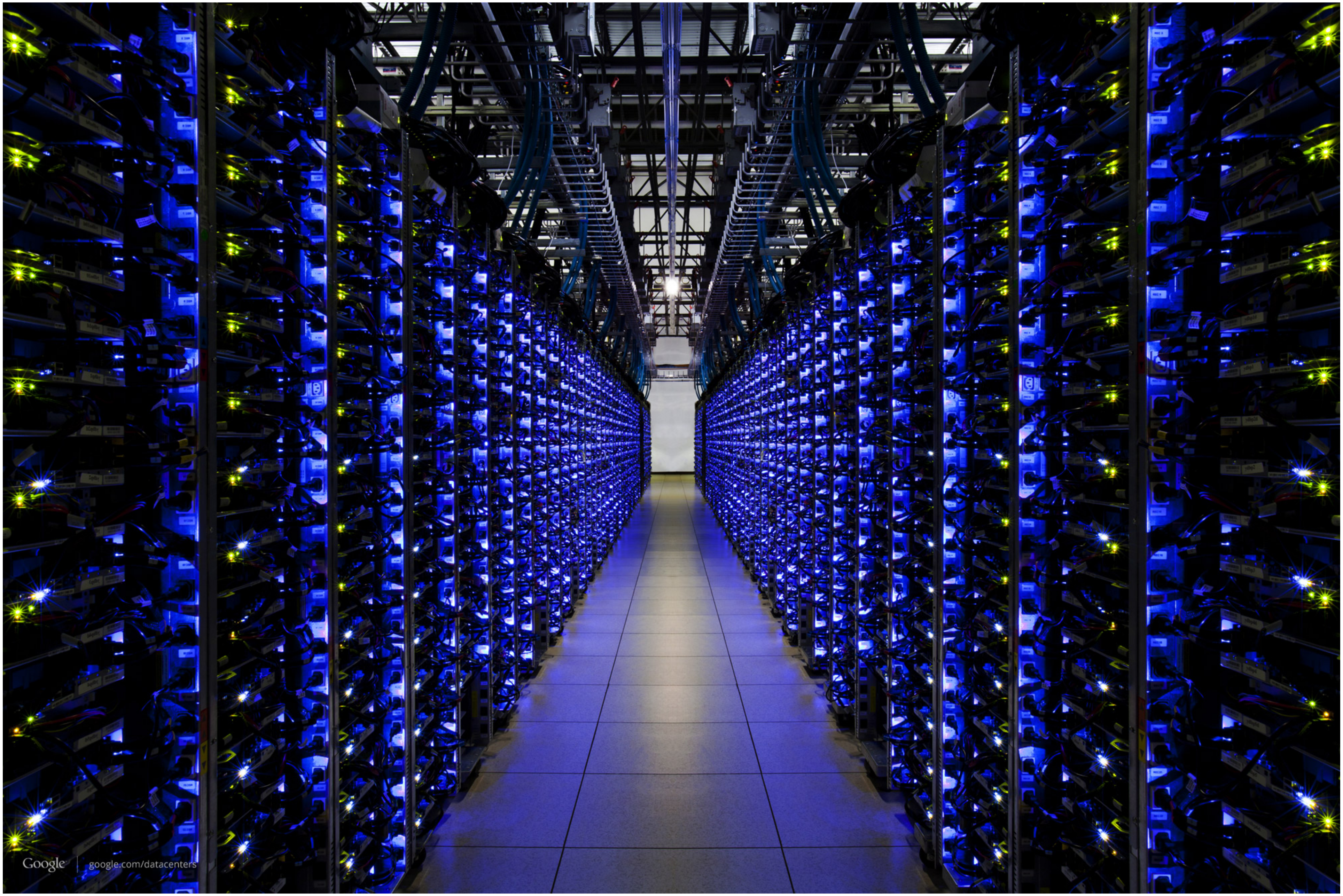
CNN

ebay

Expedia

The New York Times

GEORGE MASON UNIVERSITY

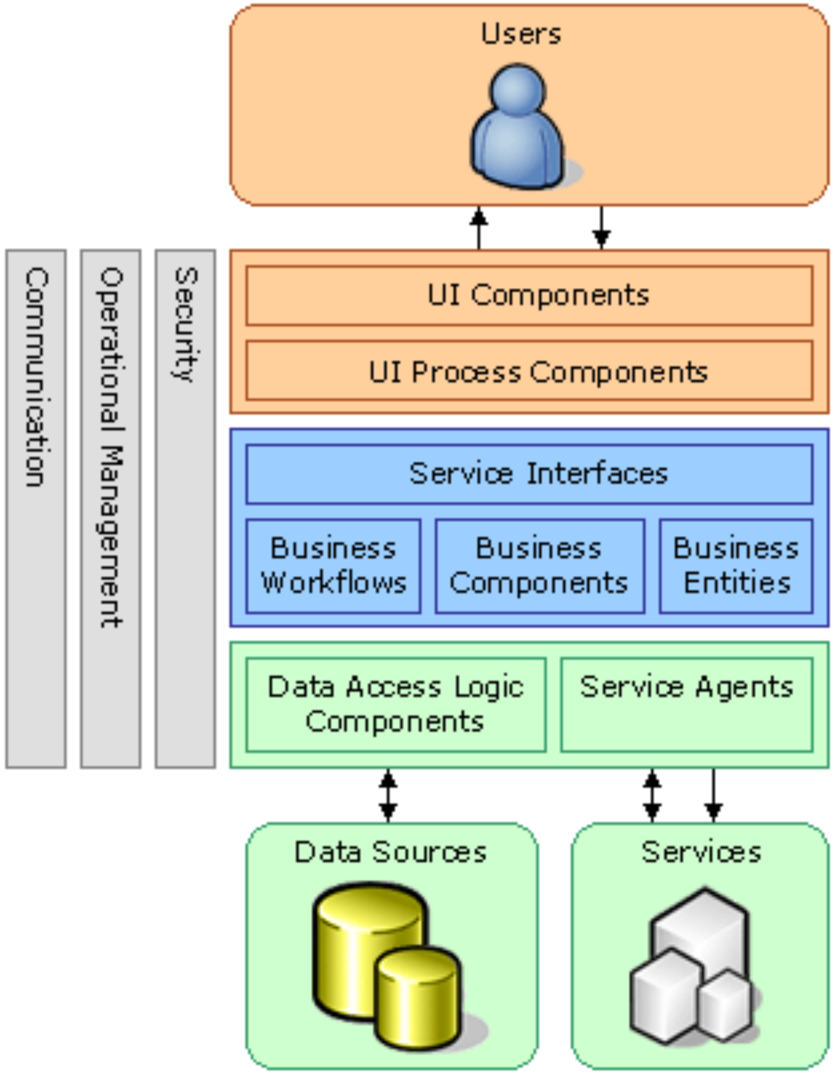

Google
google.com/datacenters

# Large Number of Configurations

- Complex middleware and database systems have a very large number of configurable parameters.

| <u>**Web Server (IIS 5.0)**</u> | <u>**Application Server (Tomcat 4.1)**</u> | <u>**Database Server (SQL Server 7.0)**</u> |
|---|---|---|
| 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 | | |

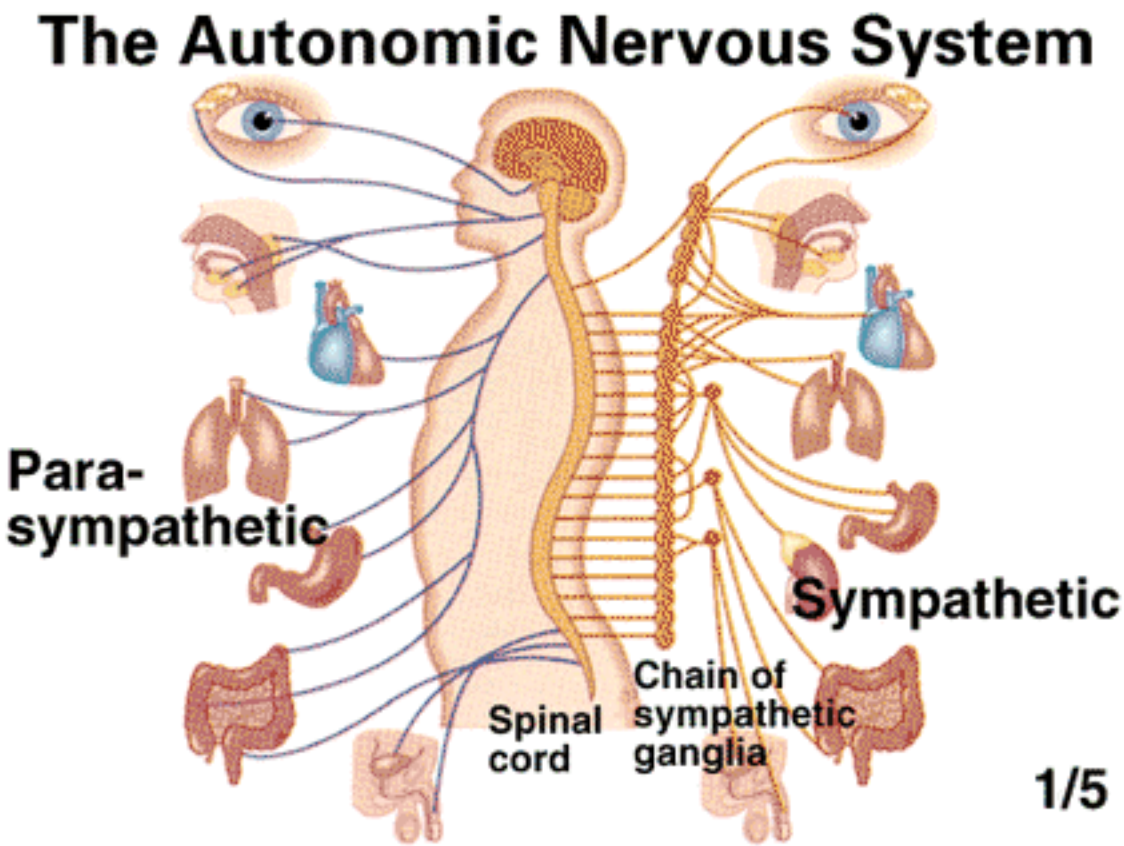
# Layered Software Architecture



# SELF-MANAGED SYSTEMS

# aka AUTONOMIC SYSTEMS

# Autonomic Computing



# Autonomic Systems

- Can manage themselves given high-level objectives expressed in term of service-level objectives or utility functions.

# Autonomic Systems

- Can manage themselves given high-level objectives expressed in term of service-level objectives or utility functions.
  - Average response time < 1.0 sec

# Autonomic Systems

- Can manage themselves given high-level objectives expressed in term of service-level objectives or utility functions.
  - Average response time < 1.0 sec
  - Response time of 95% of transactions ≤ 0.5 sec

# Autonomic Systems

- Can manage themselves given high-level objectives expressed in term of service-level objectives or utility functions.
  - Average response time < 1.0 sec
  - Response time of 95% of transactions ≤ 0.5 sec
  - Search engine throughput ≥ 4600 queries/sec

# Autonomic Systems

- Can manage themselves given high-level objectives expressed in term of service-level objectives or utility functions.
  - Average response time < 1.0 sec
  - Response time of 95% of transactions ≤ 0.5 sec
  - Search engine throughput ≥ 4600 queries/sec
  - Availability of the e-mail portal ≥ 99.99%.
  - Percentage of phishing e-mails filtered by the e-mail portal ≥ 90%

# Why SLOs are important

- 1 second of additional page load time would cost Amazon $1.6 billion in sales per year.

# Why SLOs are important

- 1 second of additional page load time would cost Amazon $1.6 billion in sales per year.
- Conversions decline sharply when load times jump from 1 to 4 sec. For every second of improvement Walmart experienced a 2% conversion increase.

# Why SLOs are important

- 1 second of additional page load time would cost Amazon $1.6 billion in sales per year.
- Conversions decline sharply when load times jump from 1 to 4 sec. For every second of improvement Walmart experienced a 2% conversion increase.
- A lag time of 400msec results in 440 million abandoned sessions/month and a massive loss in ad revenue for Google.

GEORGE MASON UNIVERSITY

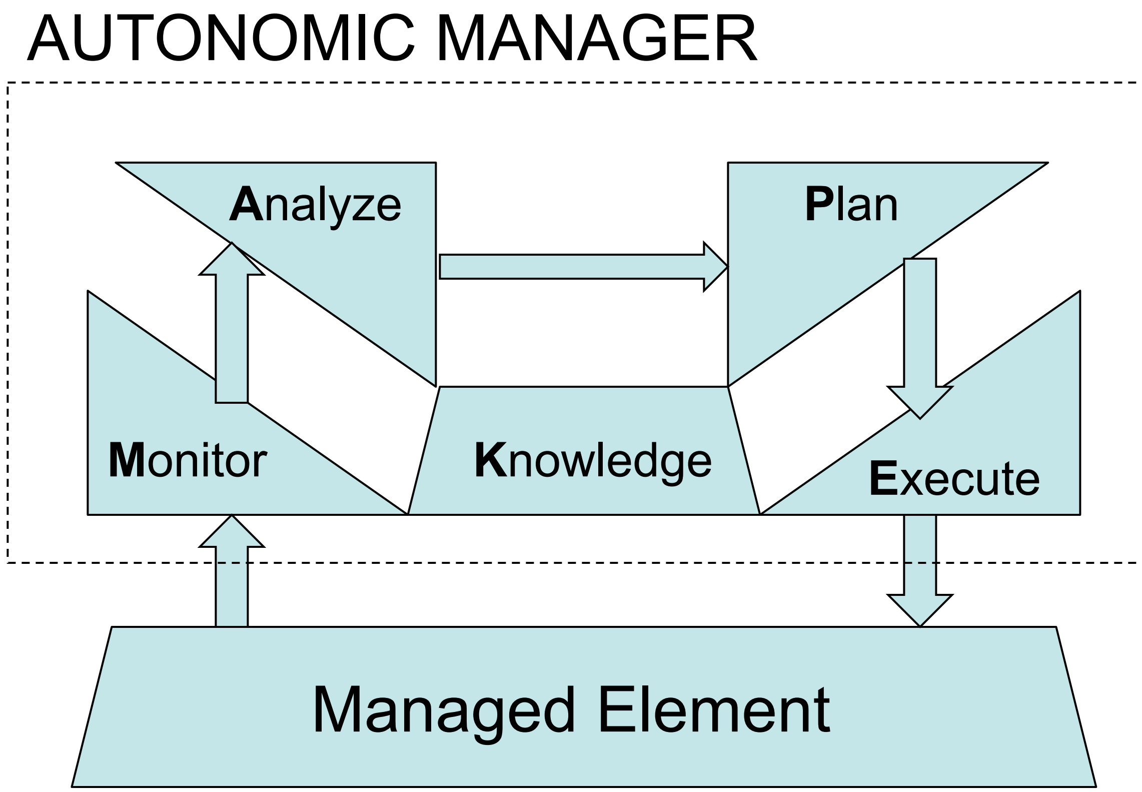
# Self-managed Systems

- Self-managing
  - Self-configuring
  - Self-optimizing
  - Self-healing
  - Self-protecting
- Self-* systems (aka autonomic systems)

# Self-managed Systems

- Self-managing
  - Self-configuring
  - Self-optimizing
  - Self-healing
  - Self-protecting
- Self-managed systems

# IBM's MAPE-K Model for AC



# Motivation for AC

- "…main obstacle to further progress in IT is a looming software complexity crisis." (from an IBM manifesto, Oct. 2001).
  - Tens of millions of lines of code

# Motivation for AC

- "…main obstacle to further progress in IT is a looming software complexity crisis." (from an IBM manifesto, Oct. 2001).
  - Tens of millions of lines of code
  - Skilled IT professionals required to install, configure, tune, and maintain.

# Motivation for AC

- "…main obstacle to further progress in IT is a looming software complexity crisis." (from an IBM manifesto, Oct. 2001).
  - Tens of millions of lines of code
  - Skilled IT professionals required to install, configure, tune, and maintain.
  - Need to integrate many heterogeneous systems

# Motivation for AC

- "…main obstacle to further progress in IT is a looming software complexity crisis." (from an IBM manifesto, Oct. 2001).
  - Tens of millions of lines of code
  - Skilled IT professionals required to install, configure, tune, and maintain.
  - Need to integrate many heterogeneous systems
  - Limit of human capacity being achieved

# Motivation for AC (cont'd)

- Harder to anticipate interactions between components at design time:
  - Need to defer decisions to run time

# Motivation for AC (cont’d)

- Harder to anticipate interactions between components at design time:
  - Need to defer decisions to run time
- Computer systems are becoming too massive, complex to be managed even by the most skilled IT professionals

# Motivation for AC (cont'd)

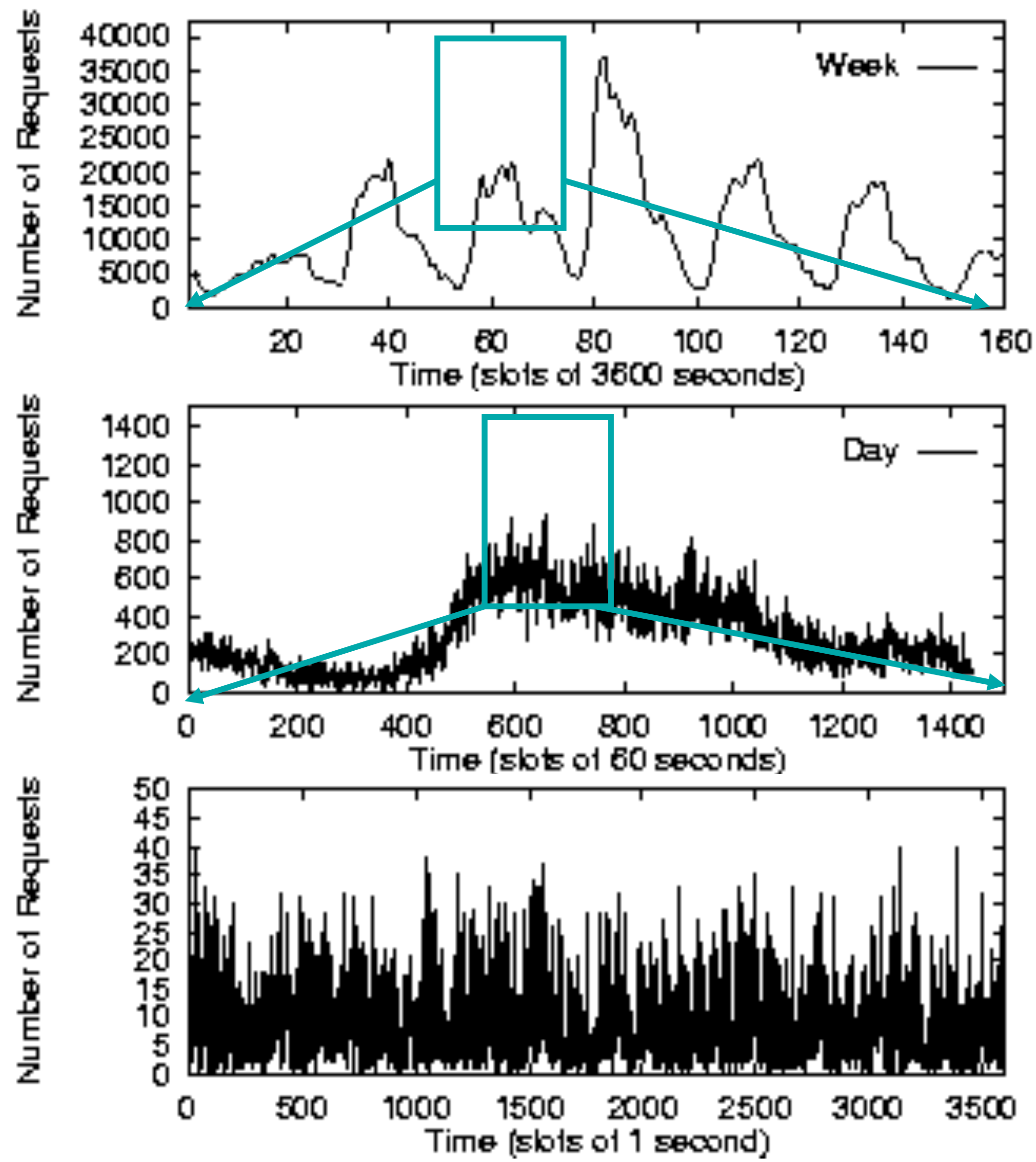
- Harder to anticipate interactions between components at design time:
  - Need to defer decisions to run time
- Computer systems are becoming too massive, complex to be managed even by the most skilled IT professionals
- The workload and environment conditions tend to change very rapidly with time

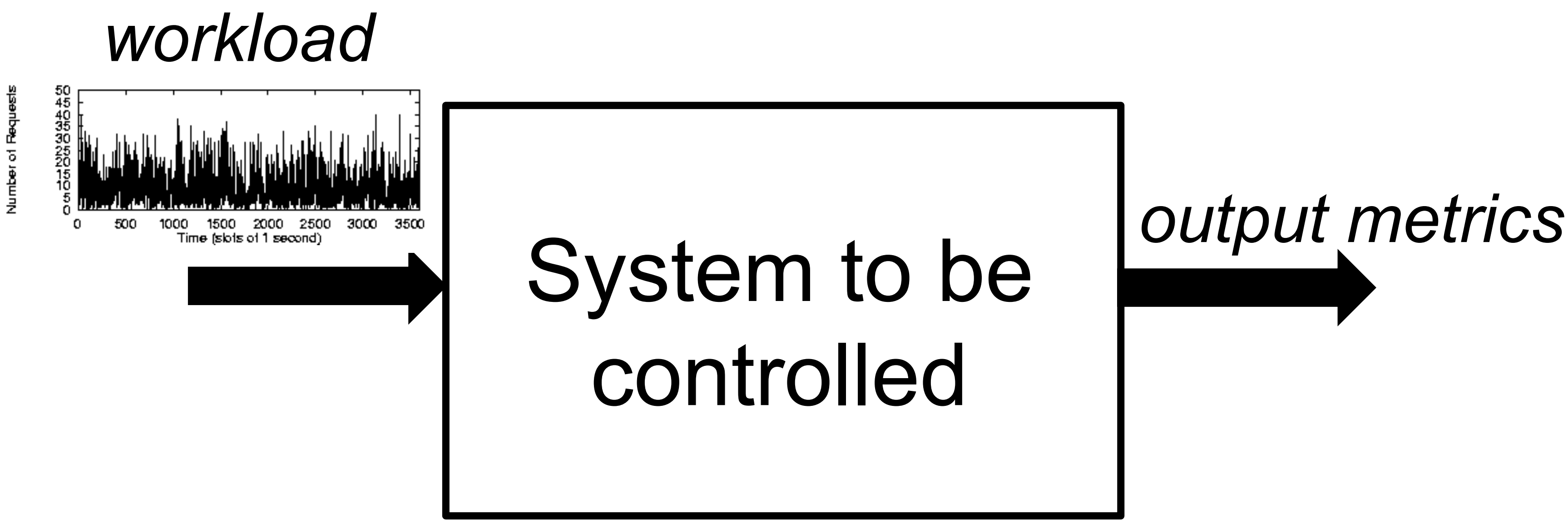
# Multi-scale time workload variation of a Web Server

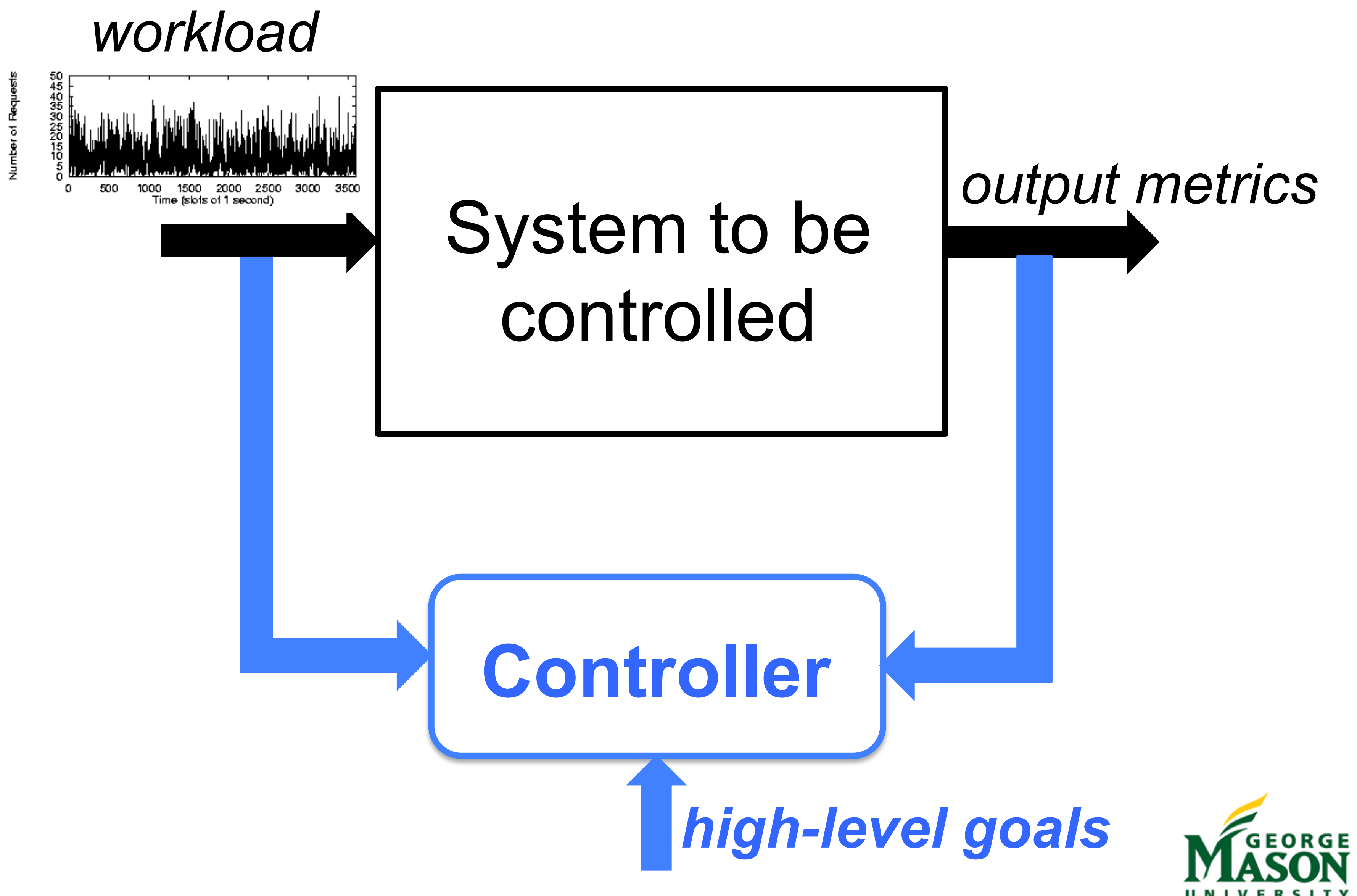
3600 sec

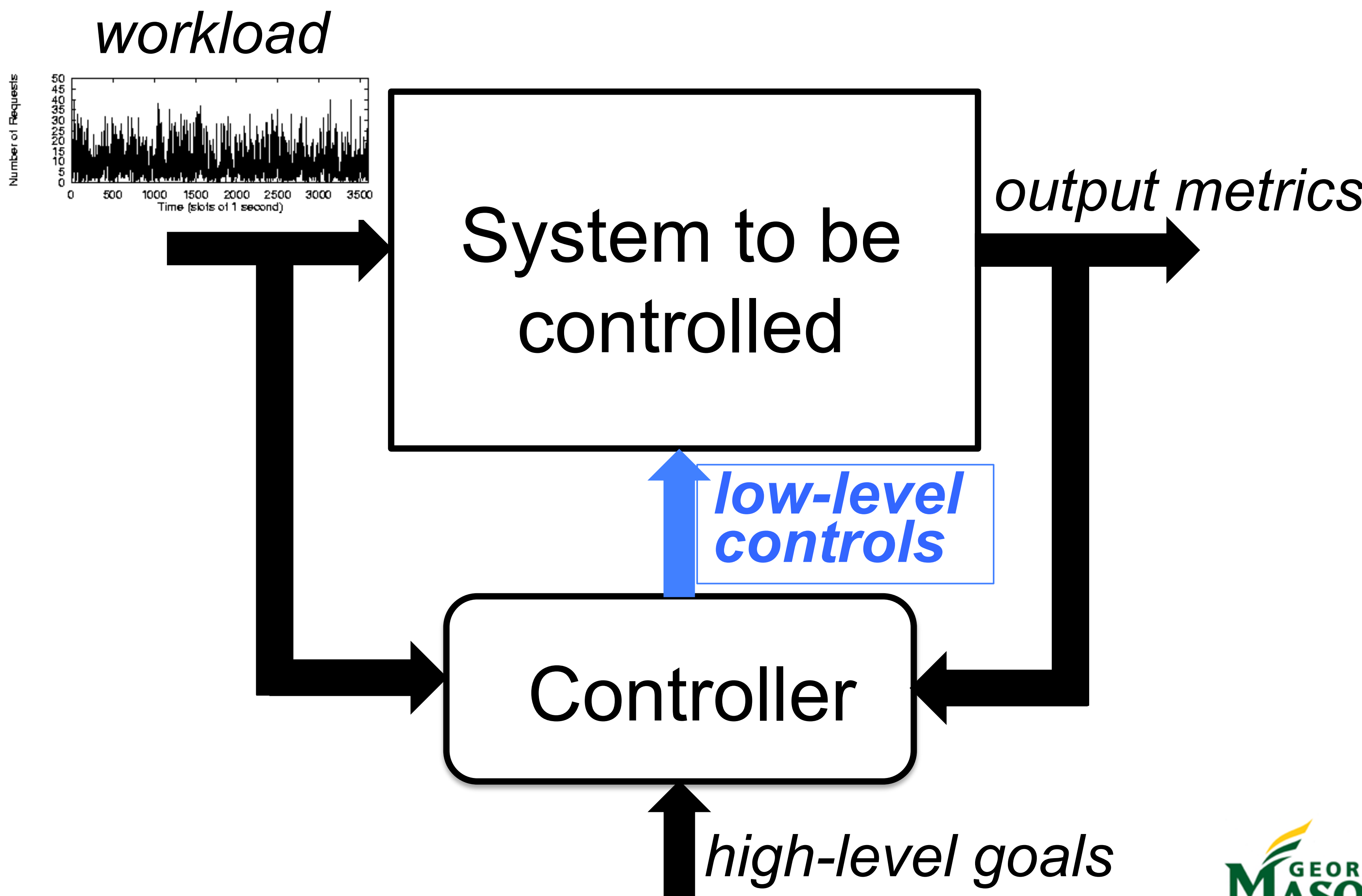
60 sec

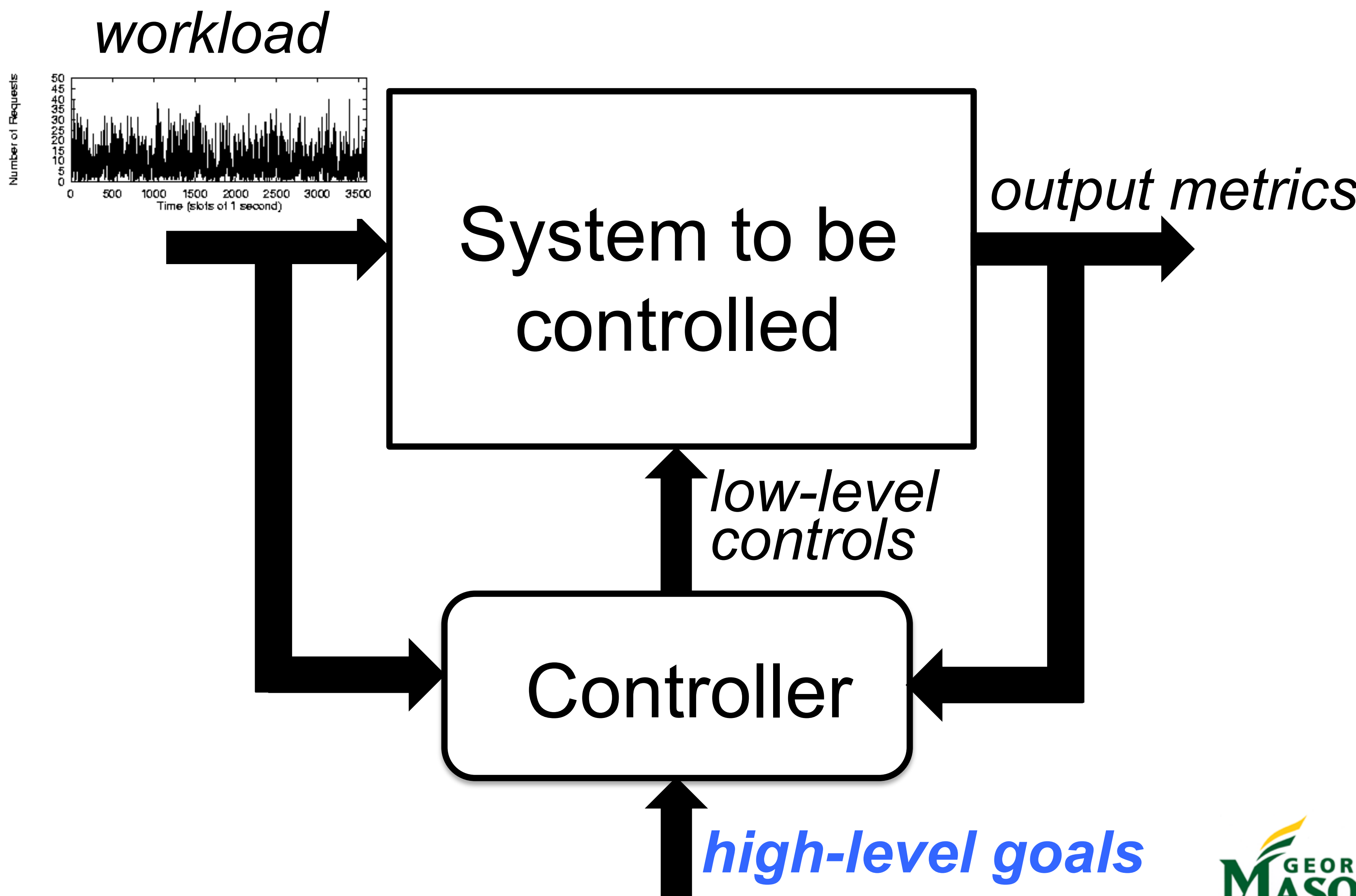
1 sec



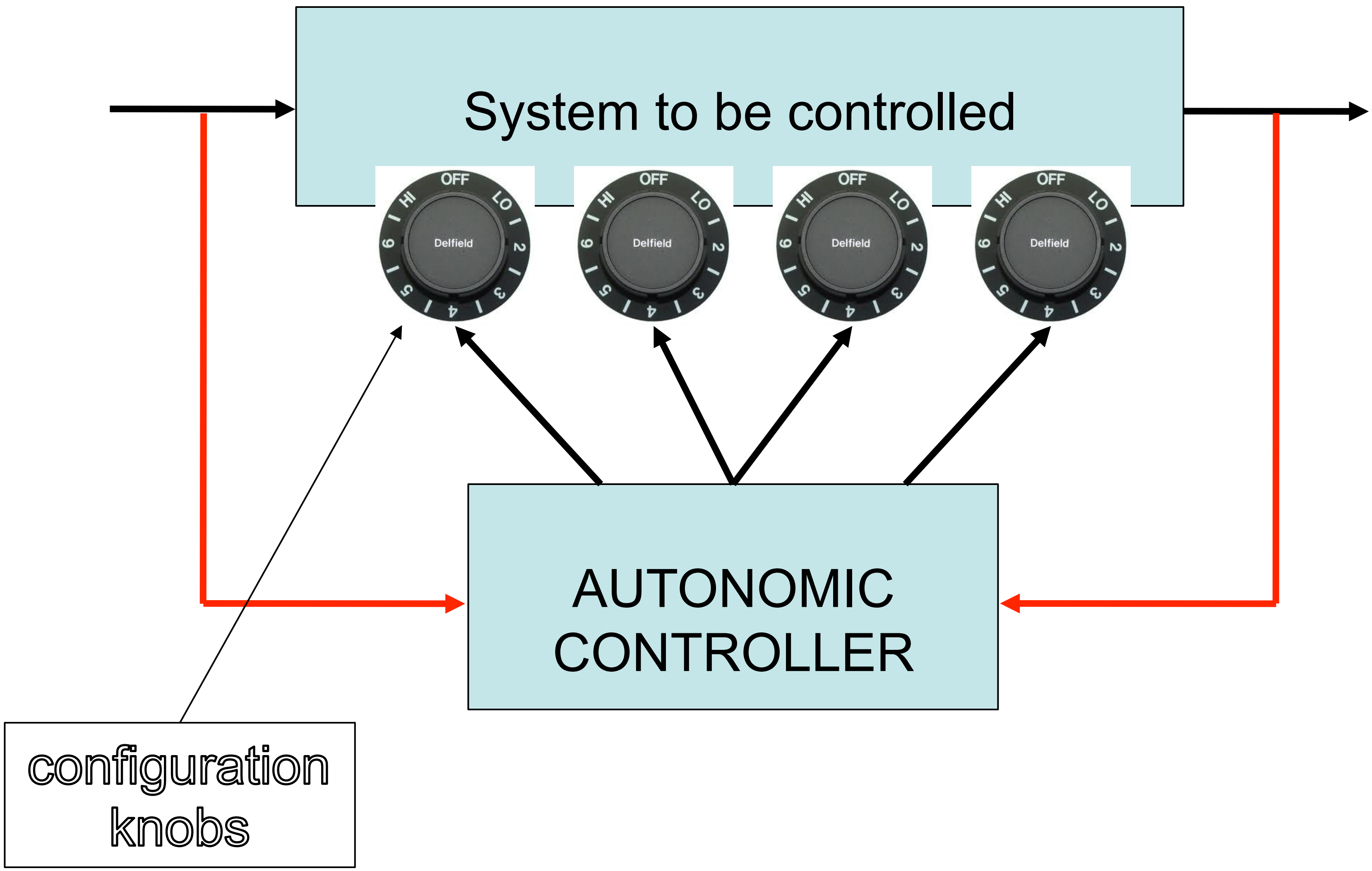

workload
Number of Requests
0 5 10 15 20 25 30 35 40 45 50
0 500 1000 1500 2000 2500 3000 3500
Time (slots of 1 second)
System to be controlled
output metrics


workload
Number of Requests
Time (slots of 1 second)
System to be controlled
output metrics
Controller
high-level goals
GEORGE MASON UNIVERSITY

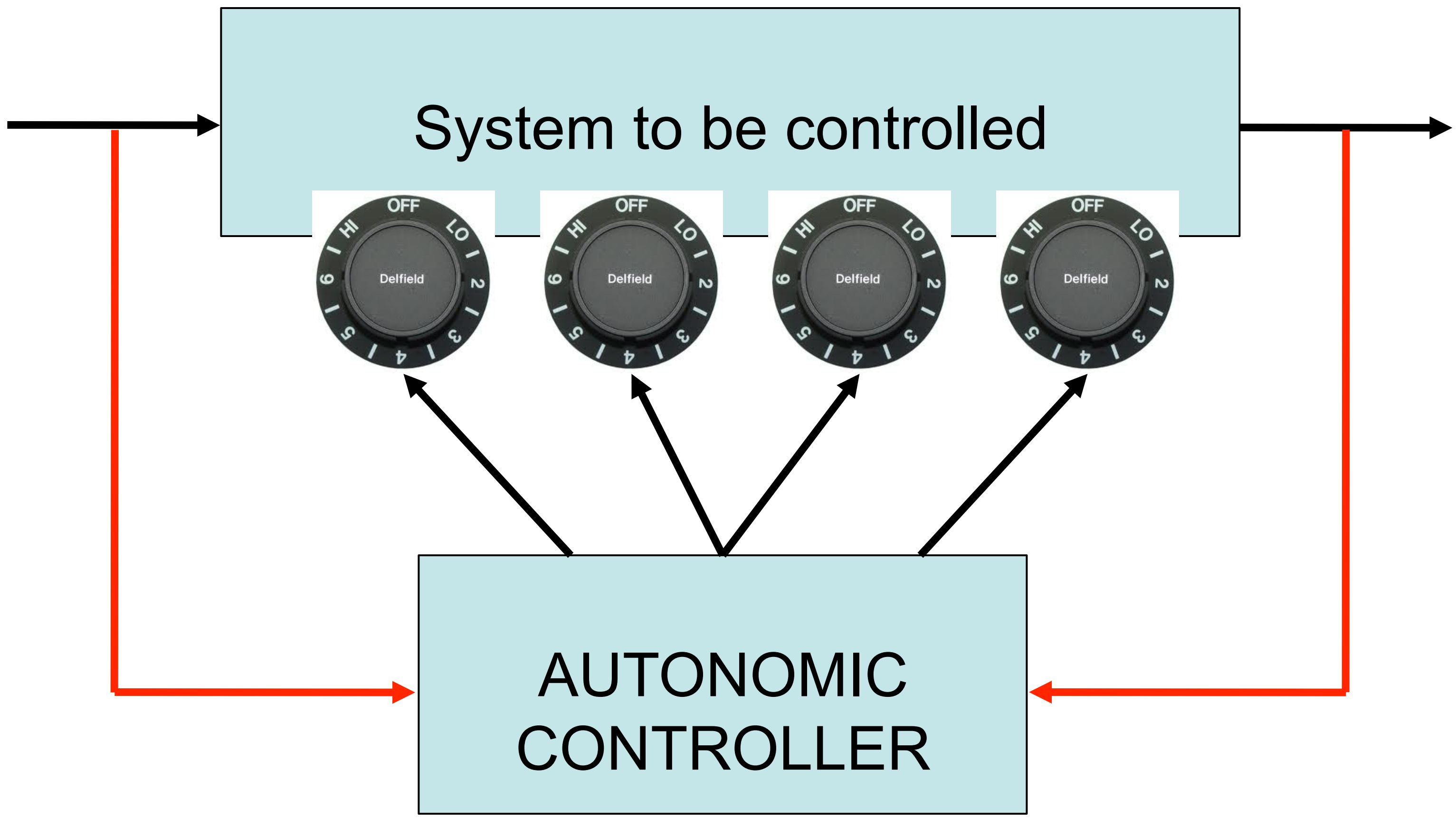

workload
Number of Requests
50
45
40
35
30
25
20
15
10
5
0
0
500
1000
1500
2000
2500
3000
3500
Time (slots of 1 second)
System to be controlled
output metrics
low-level controls
Controller
high-level goals


workload
Number of Requests
Time (slots of 1 second)
System to be controlled
output metrics
low-level controls
Controller
high-level goals

# Autonomic Controller

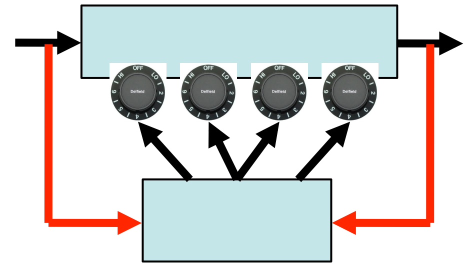


# Autonomic Controller




How does the AC know the output of the system for a given combination of the knobs?

# Autonomic Controller

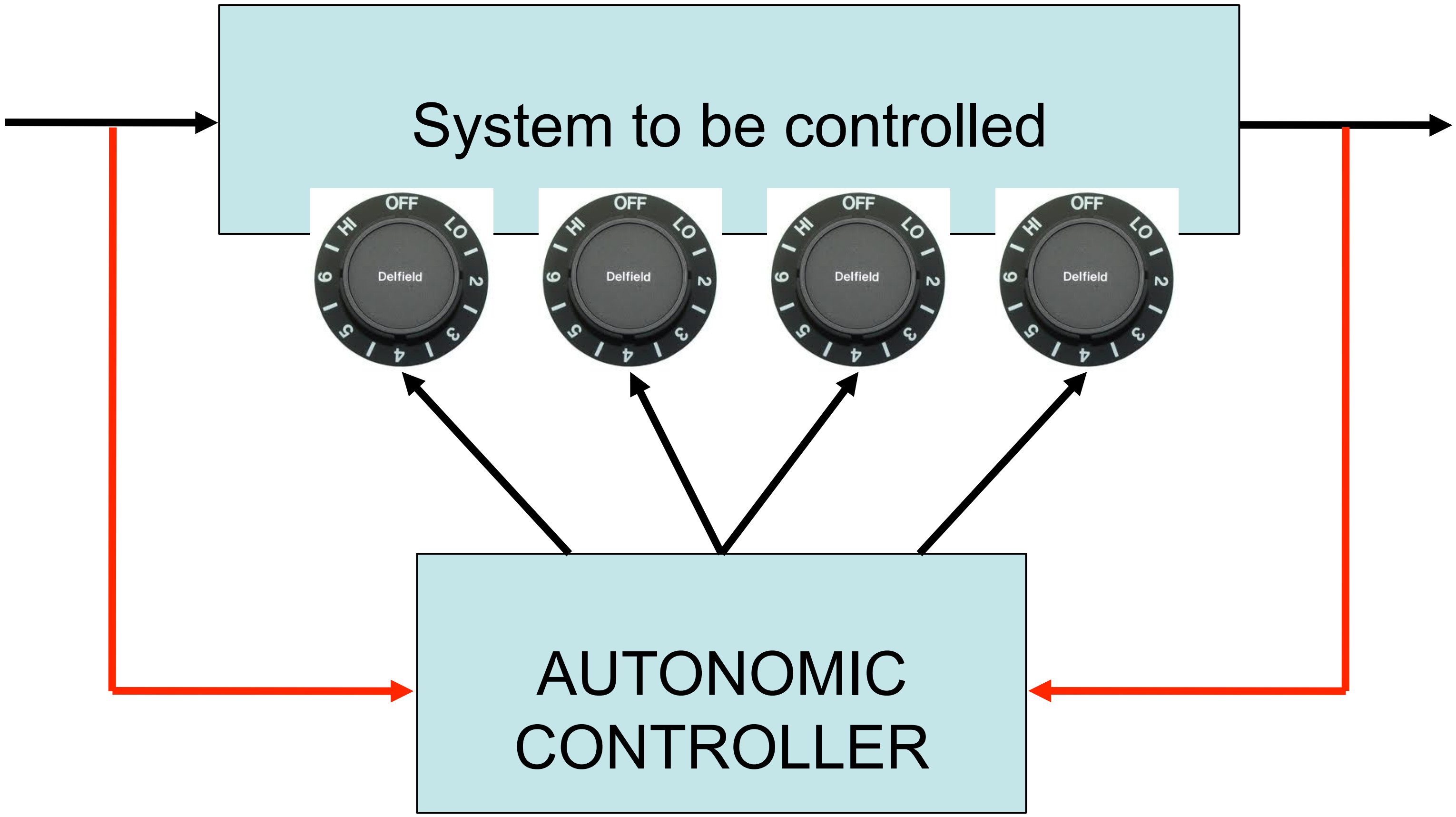


How does the AC know the output of the system for a given combination of the knobs?

$$S_{out} = f(k_1, k_2, \ldots, k_n, S_{input})$$

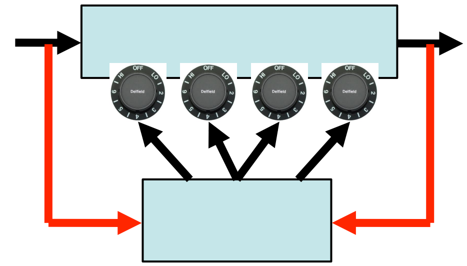
The function *f* can be obtained by a model or can be learned by the AC controller by observing system inputs and outputs.

# Autonomic Controller




What is the objective of the AC when determining a new set of knobs (i.e., configuration) for the system?

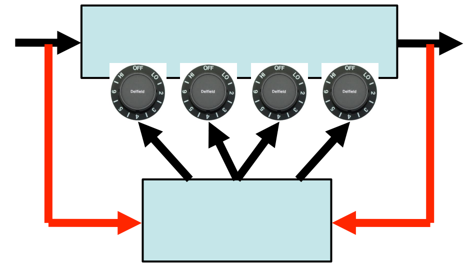
# Autonomic Controller



What is the objective of the AC when determining a new set of knobs (i.e., configuration) for the system?

- The AC may want to maximize/minimize a performance metric:
  - Minimize response time
  - Maximize availability
  - Maximize throughput
  - Minimize energy consumption

# Autonomic Controller

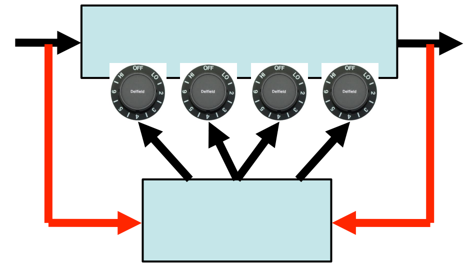


What is the objective of the AC when determining a new set of knobs (i.e., configuration) for the system?

Minimize ResponseTime = f (k1, …, kn)

Subject to
EnergyConsumed = g1 (k1, …, kn) ≤ MaxEnergy
Throughput = g2 (k1, …, kn) ≥ MinThroughput
Availability = g3 (k1, …, kn) ≥ MinAvailability

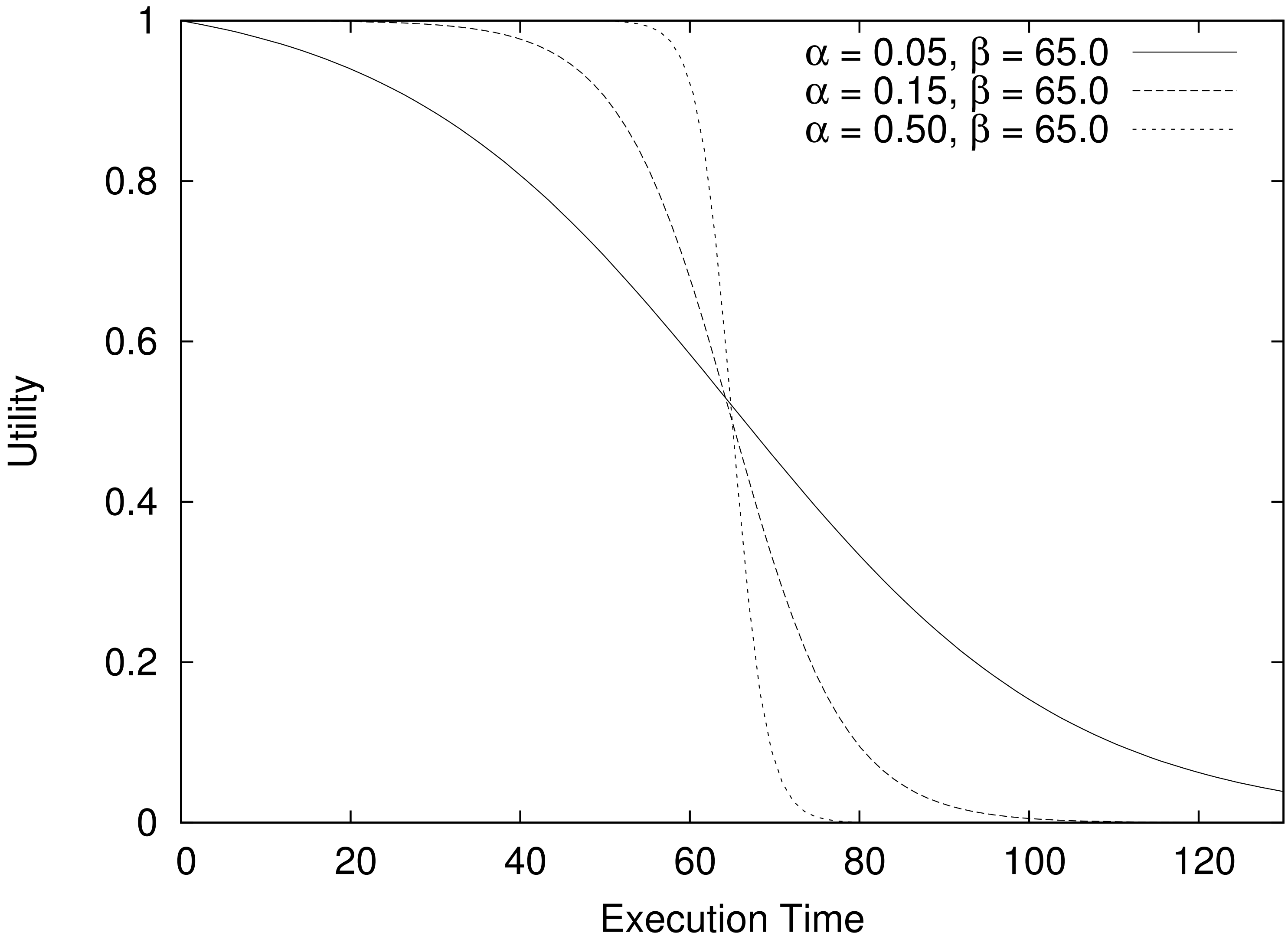
# Utility Functions and the AC



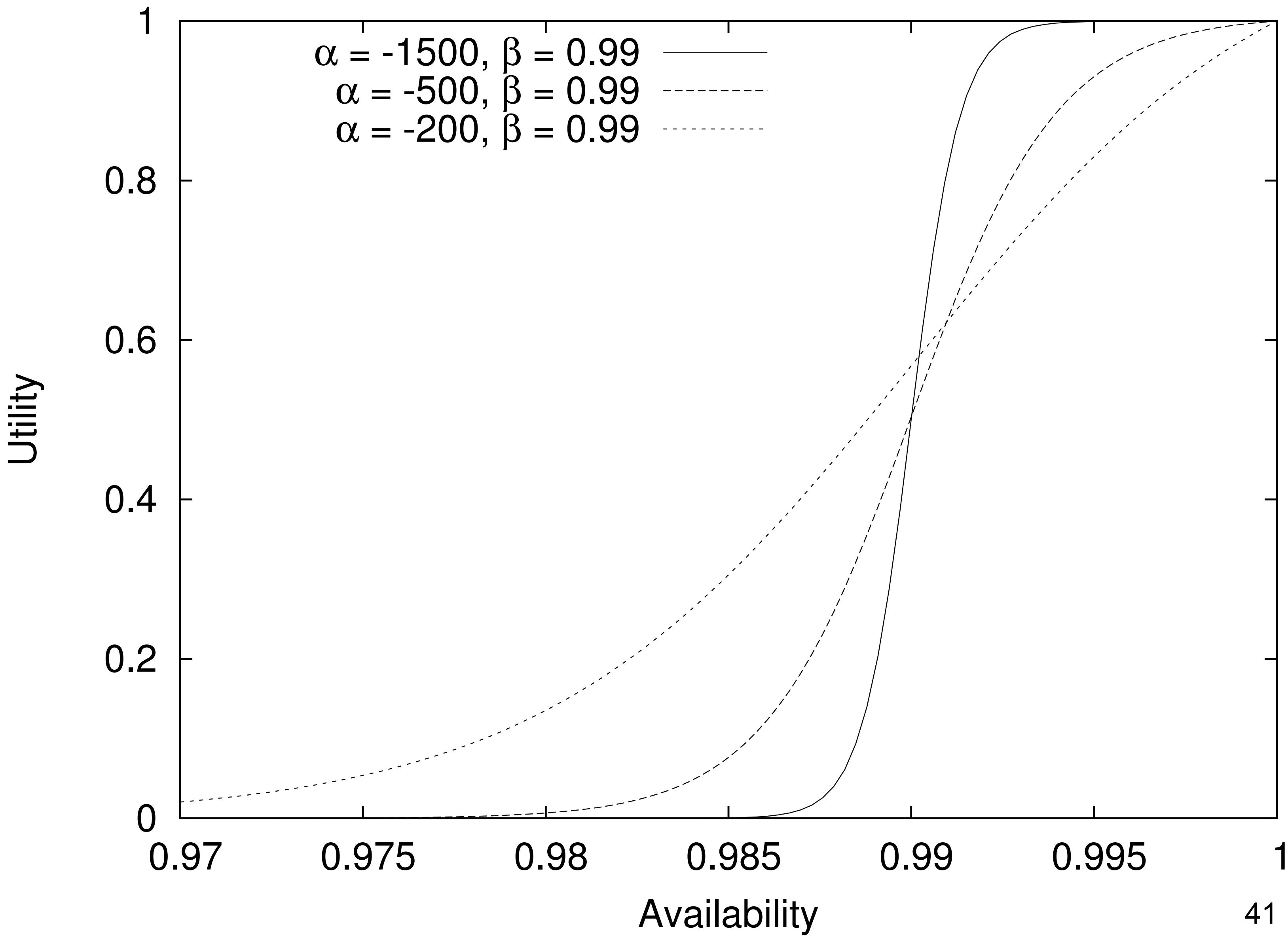
What is the objective of the AC when determining a new set of knobs (i.e., configuration) for the system?

- The AC may want to consider trade-offs between performance metrics.
- Use utility function.
  - A utility function of an attribute ***a*** indicates the usefulness of a system as a function of the value of the attribute ***a***.

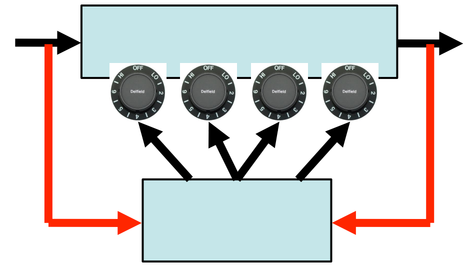
# Execution Time Utility Function



# Availability Utility Function



# Utility Functions and the AC



What if there is more than one relevant attribute?

- Specify a global utility function that is a function of the utility functions of each attribute:

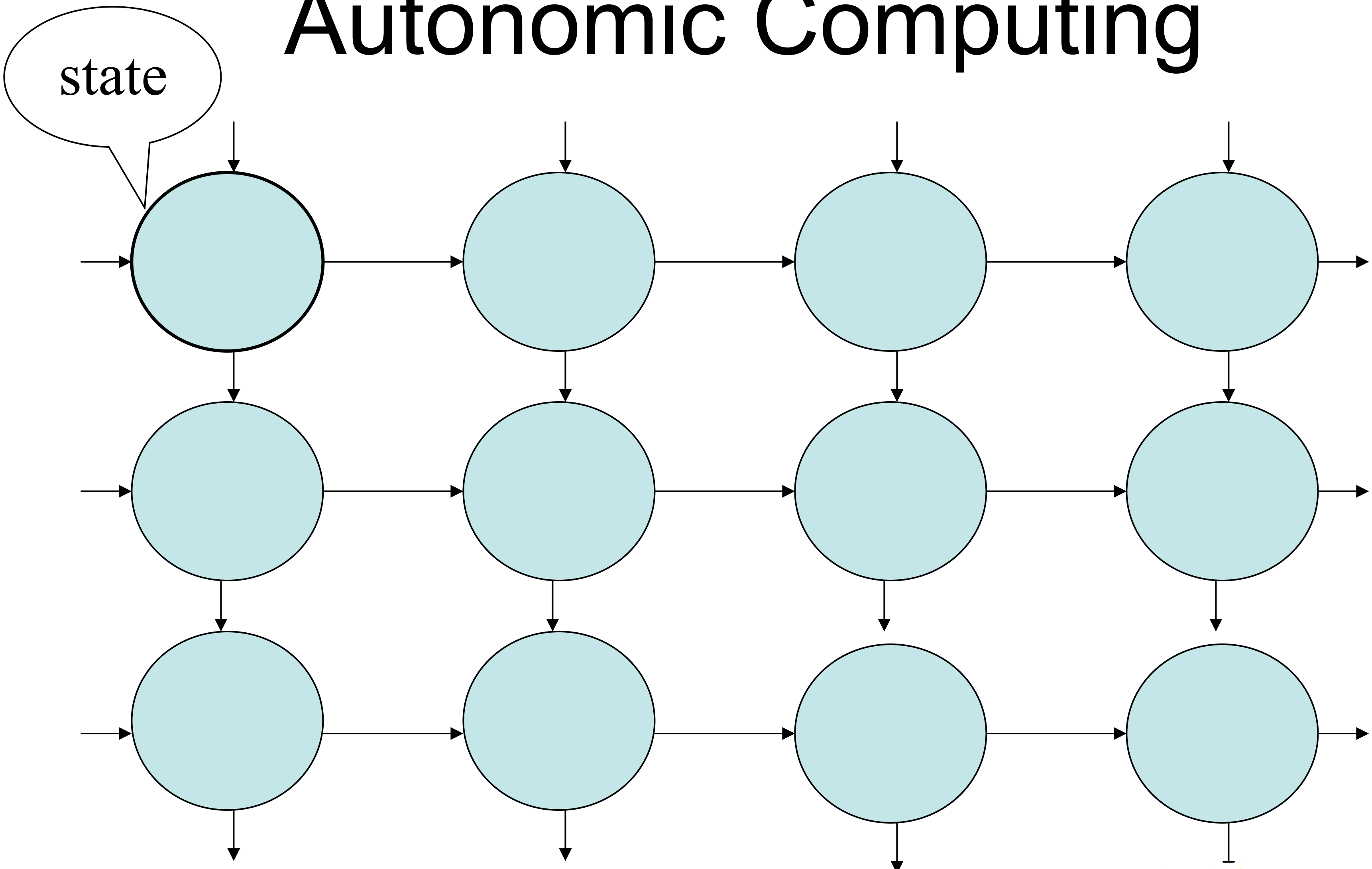
$$U_{global} = f(U_1(a_1),\ldots,U_n(a_n))$$

e.g.,

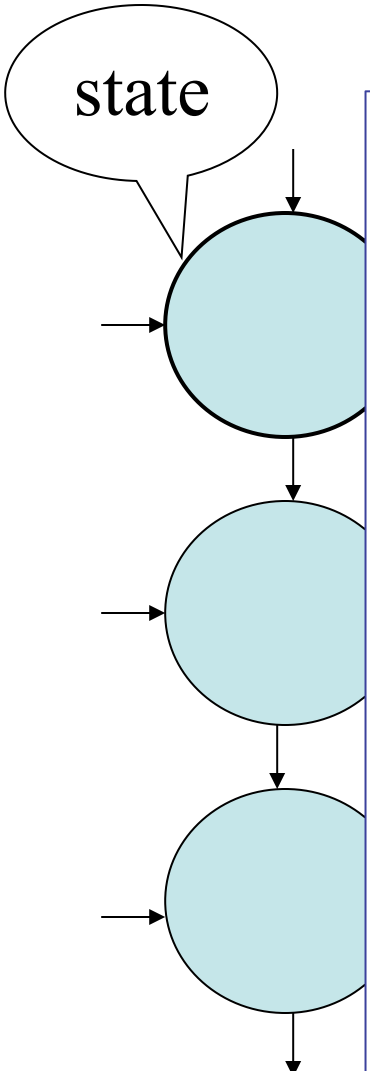
$$U_{global} = w_r U_r(R) + w_x U_x(X) + w_a U_a(a)$$

$$w_r + w_x + w_a = 1$$

# Performance Model-Based Autonomic Computing



# Performance Model-Based Autonomic Computing




State: e.g., set of configuration parameters

Value: e.g., QoS metric, utility function value

Goal: find state that optimizes the value subject to constraints

- State space is typically large
- Objective function does not have a closed form

Use performance models to compute value at each state.

Use combinatorial search techniques to find near-optimal solution.

# DYNAMIC ALLOCATION OF CPU SHARES TO VMs

- "Autonomic Virtualized Environments," M.N. Bennani and D.A. Menasce, *IEEE International Conference on Autonomic and Autonomous Systems,* July 19-21, 2006, Silicon Valley, CA, USA.

# CPU Allocation Problem for Autonomic Virtualized Environments

- Existing systems allow for manual allocation of CPU resources to VMs using *CPU priorities* or *CPU shares*.
- Need automated mechanism for the adjustment of CPU shares of the virtual machines in order to maximize the global utility of the entire virtualized environment

# CPU Allocation Problem for Autonomic Virtualized Environments (Cont' d)

Workload 1 — $U_{1,1}$

Workload 2

....

Workload n — $U_{n,1}$

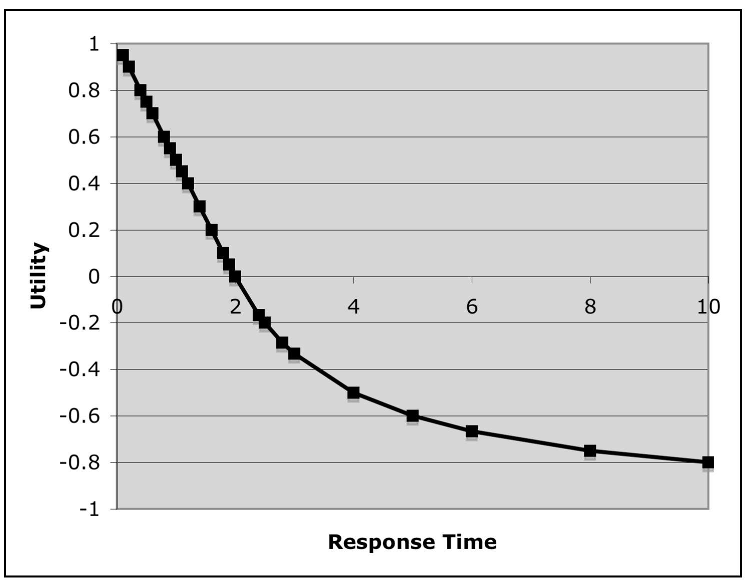
Virtual Machine 1 — $U_1$

Virtual Machine 2 — $U_2$

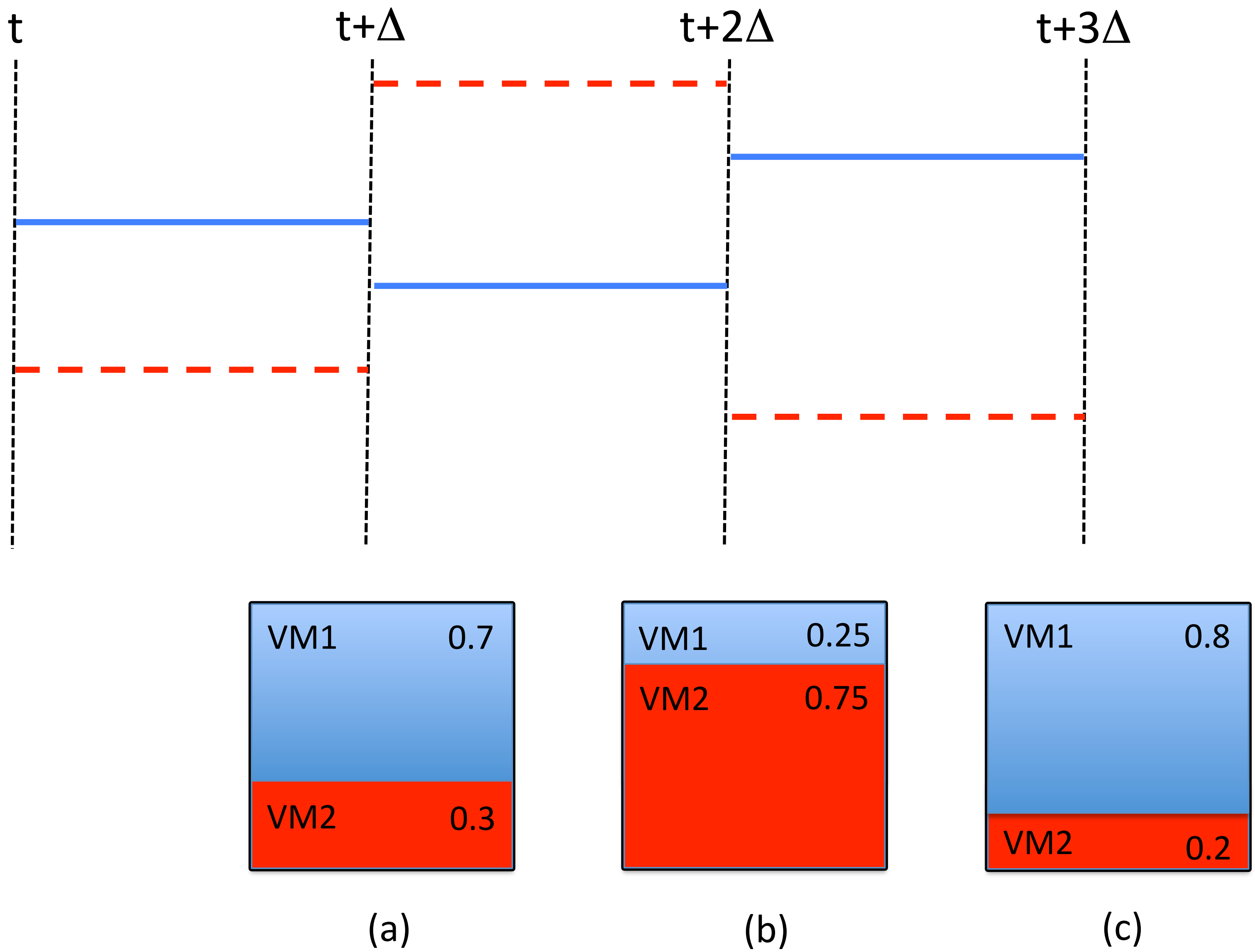
...

Virtual Machine M — $U_M$

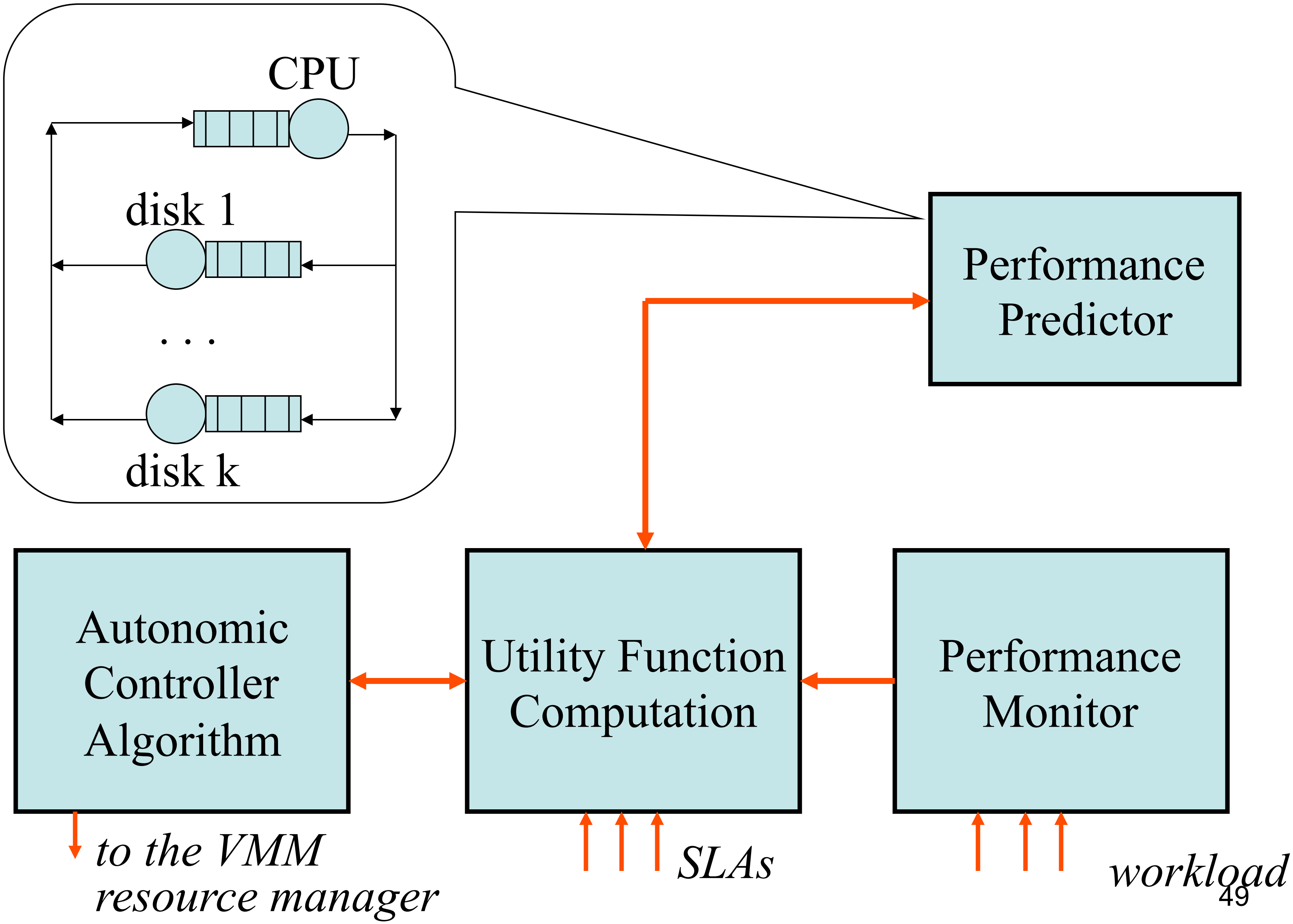
$U_g$ Virtualized Environment



# Example: CPU Shares



# Virtualization Controller Architecture



# Example: CPU Shares Allocated to VMs

*Share of CPU for VM k*

Response time at VM k:

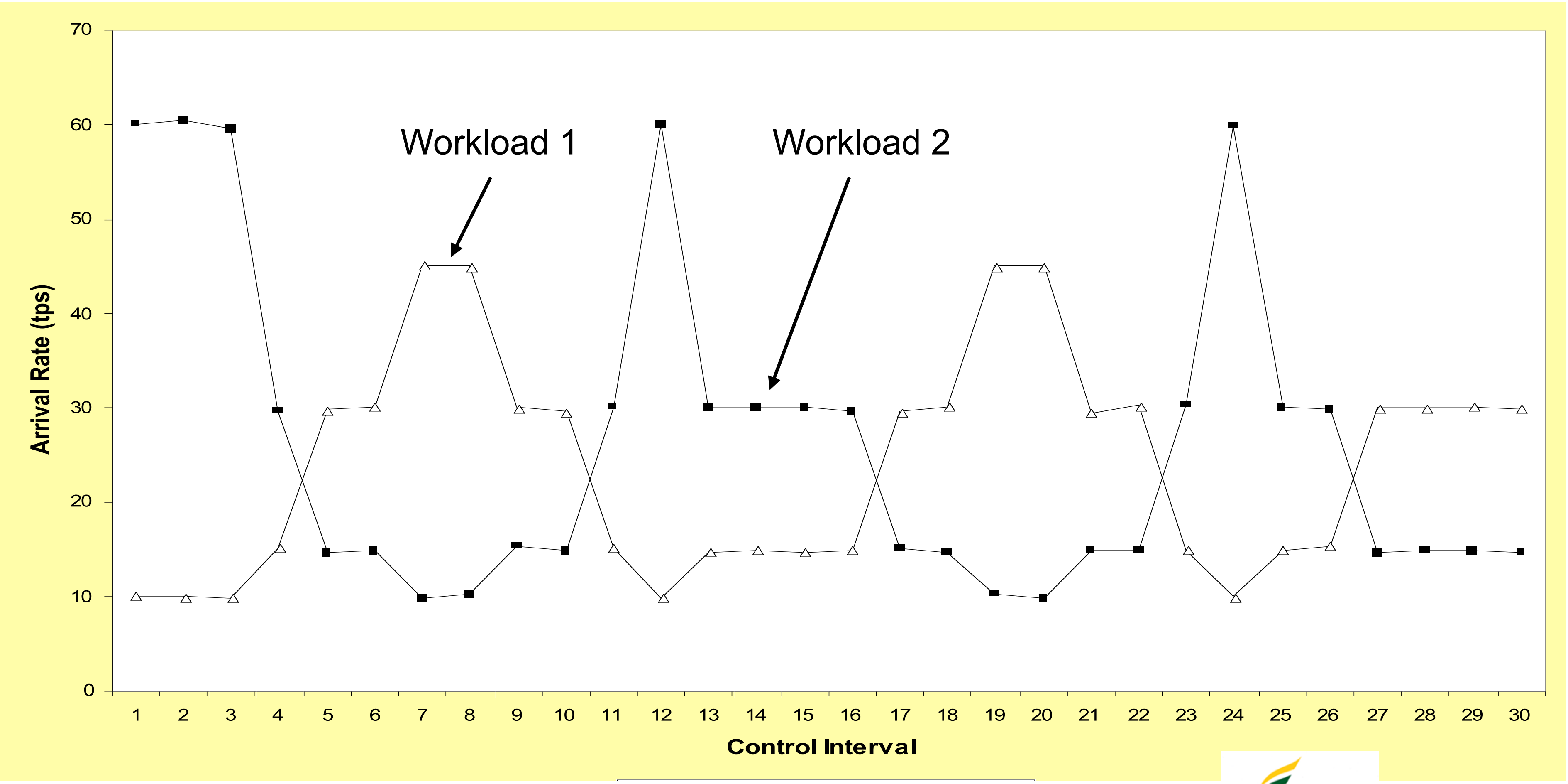
$$g_k(S(t)) = R_k(t) = \frac{D_k/C_k(t)}{1 - \mathcal{W}_k(t) D_k/C_k(t)}.$$

Utility of response time at VM k:

$$U_k(R_k(t)) = \frac{1 + e^{\alpha_k . \beta_k}}{e^{\alpha_k . \beta_k}} \frac{e^{\alpha_k(\beta_k - R_k(t))}}{1 + e^{\alpha_k(\beta_k - R_k(t))}}$$

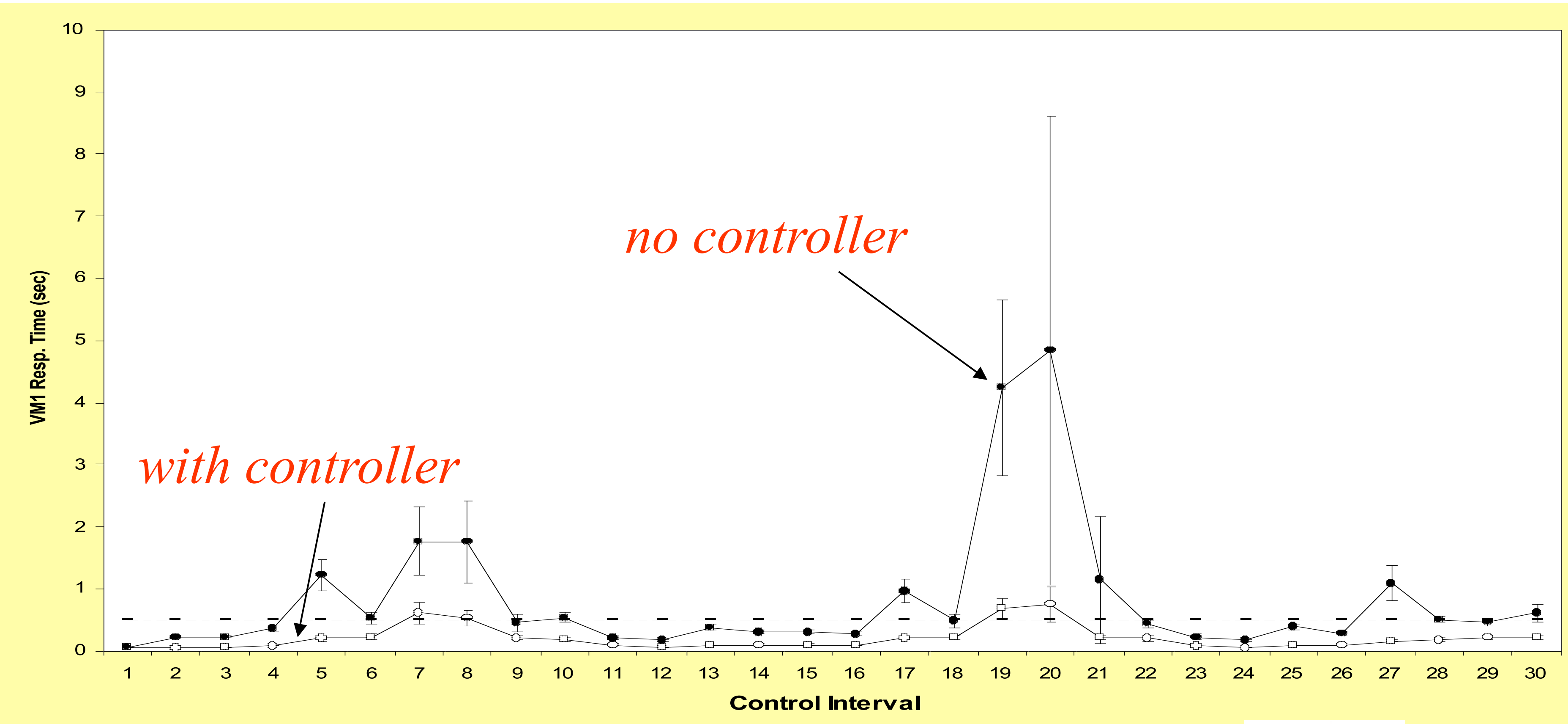
Global system utility: $$U_g(\mathbf{C}(t), \mathbf{W}(t)) = \sum_{k=1}^{K} w_k U_k(R_k(t)).$$

# CPU Shares Based Allocation: Workload Variation

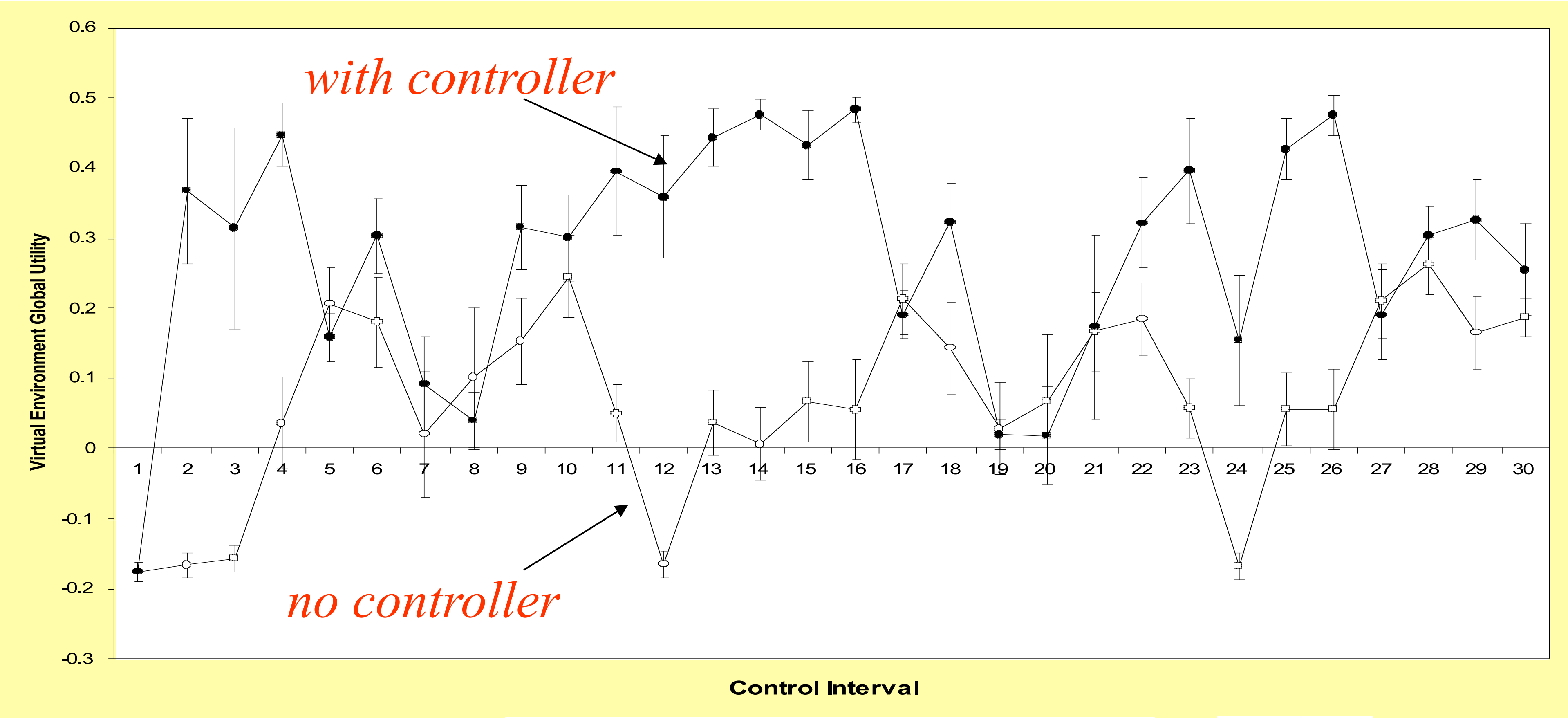


# CPU Shares Based Allocation: CPU Shares Variation

# CPU Shares Based Allocation: Response Time for VM1



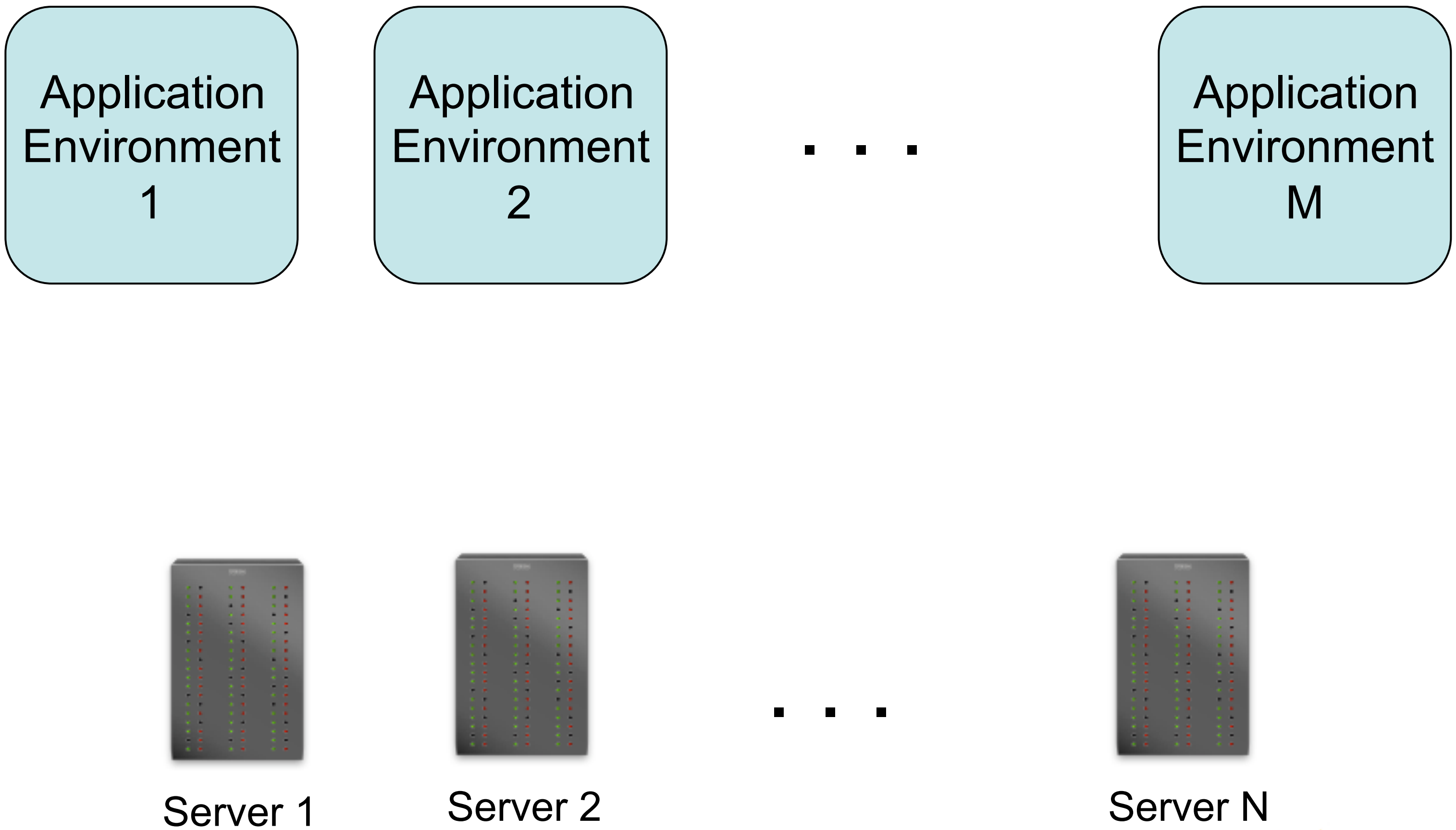
# CPU Shares Based Allocation: Global Utility



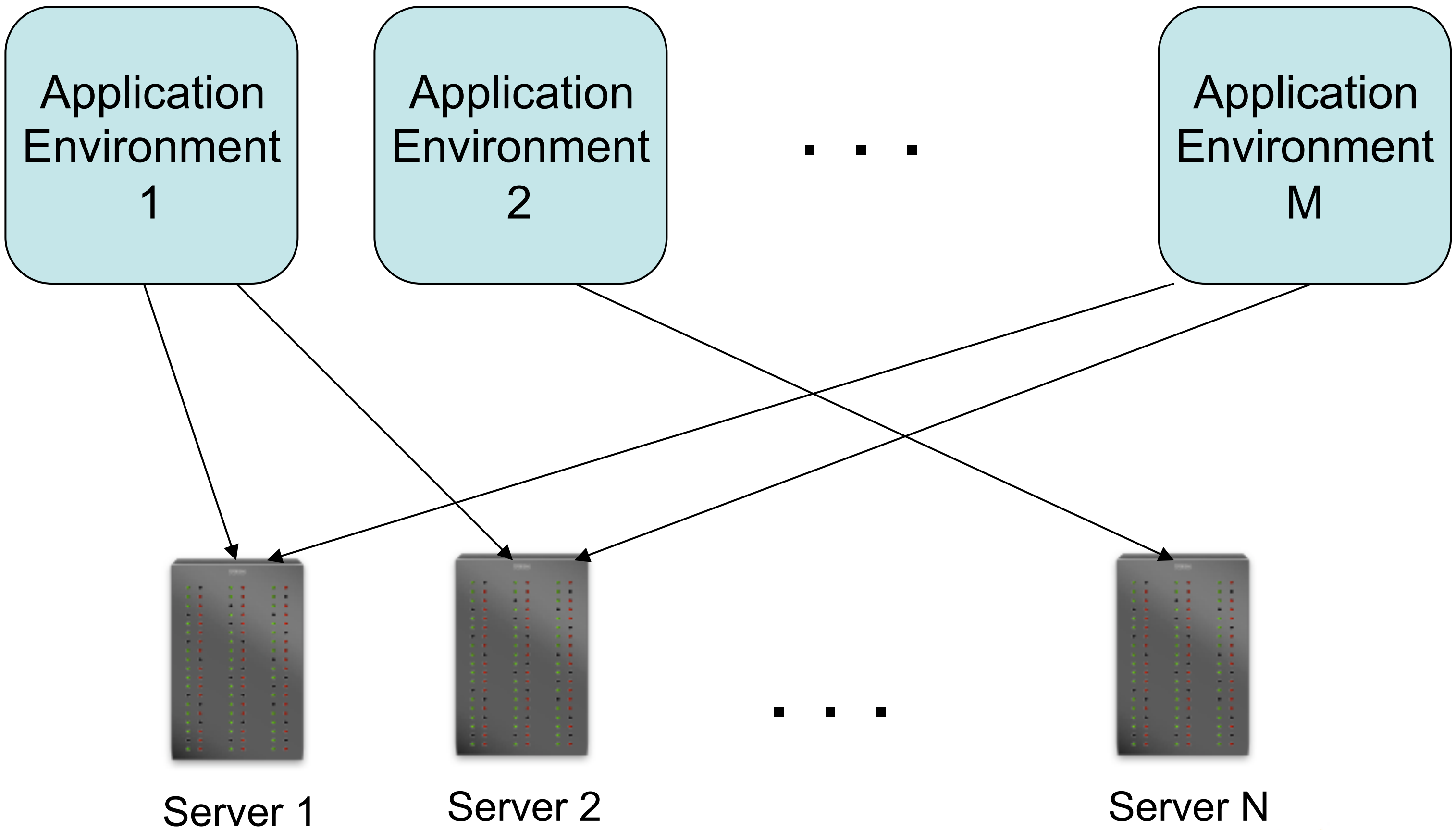
# DYNAMIC RESOURCE ALLOCATION IN INTERNET DATA CENTERS


- Dynamic Server Allocation for Autonomic Service Centers in the Presence of Failures," D.A. Menasce and M. Bennani, in the book *Autonomic Computing: Concepts, Infrastructure, and Applications*, eds. S. Hariri and M. Parashar, CRC Press.
- "Resource Allocation for Autonomic Data Centers Using Analytic Performance Models," M. Bennani and D.A. Menasce), *Proc. 2005 IEEE International Conference on Autonomic Computing,* Seattle, WA, June 13-16, 2005.
- "Assessing the Robustness of Self-Managing Computer Systems under Highly Variable Workloads," M. Bennani and D.A. Menasce, *Proc. International Conf. Autonomic Computing (ICAC-04)*, New York, NY, May 17-18, 2004.

# Dynamic Resource Allocation in Internet Data Centers



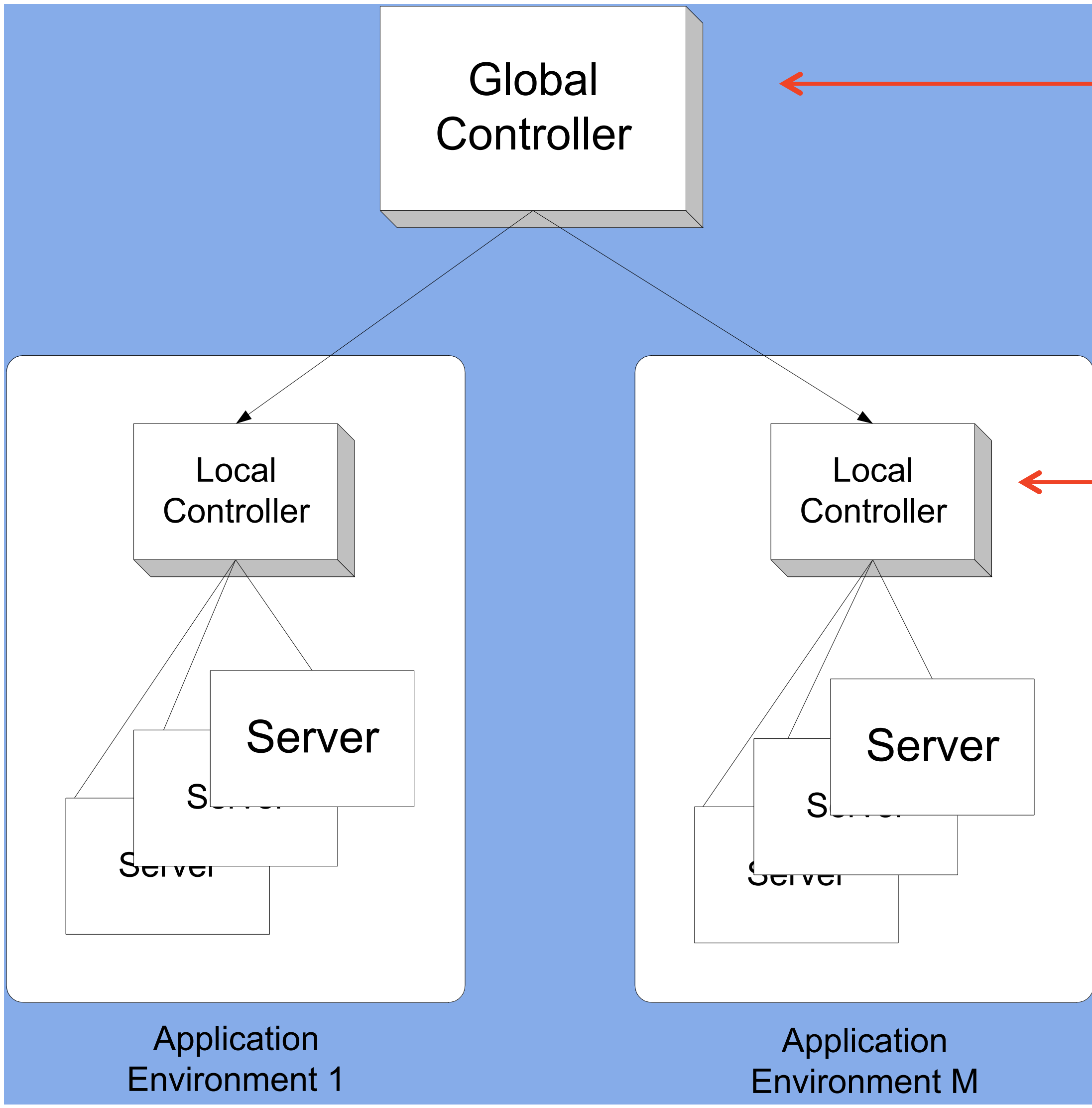
# Dynamic Resource Allocation Problem






GEORGE MASON UNIVERSITY

# Dynamic Resource Allocation Two-level Controllers




*Decides how many servers to assign to each AE.*

*Implements Global Controller's decisions.*

# Dynamic Resource Allocation Utility Function

- The global controller uses a global utility function, Ug, to assess the adherence of the overall data center performance to desired service levels objectives (SLOs)

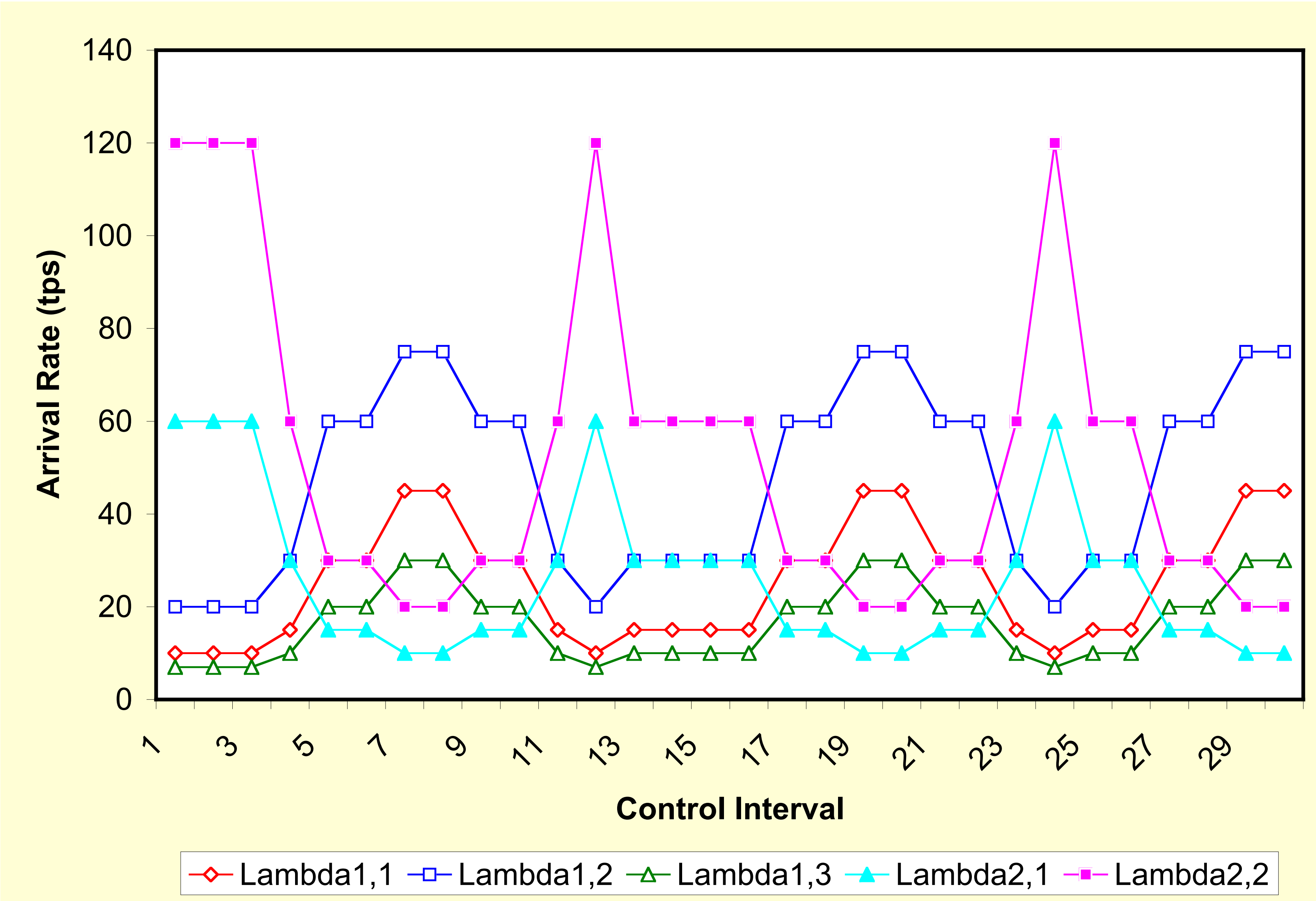
$$U_g = h(U_1, \ldots, U_M)$$

$$U_i = \sum_{s=1}^{S_i} a_{i,s} \times U_{i,s}$$

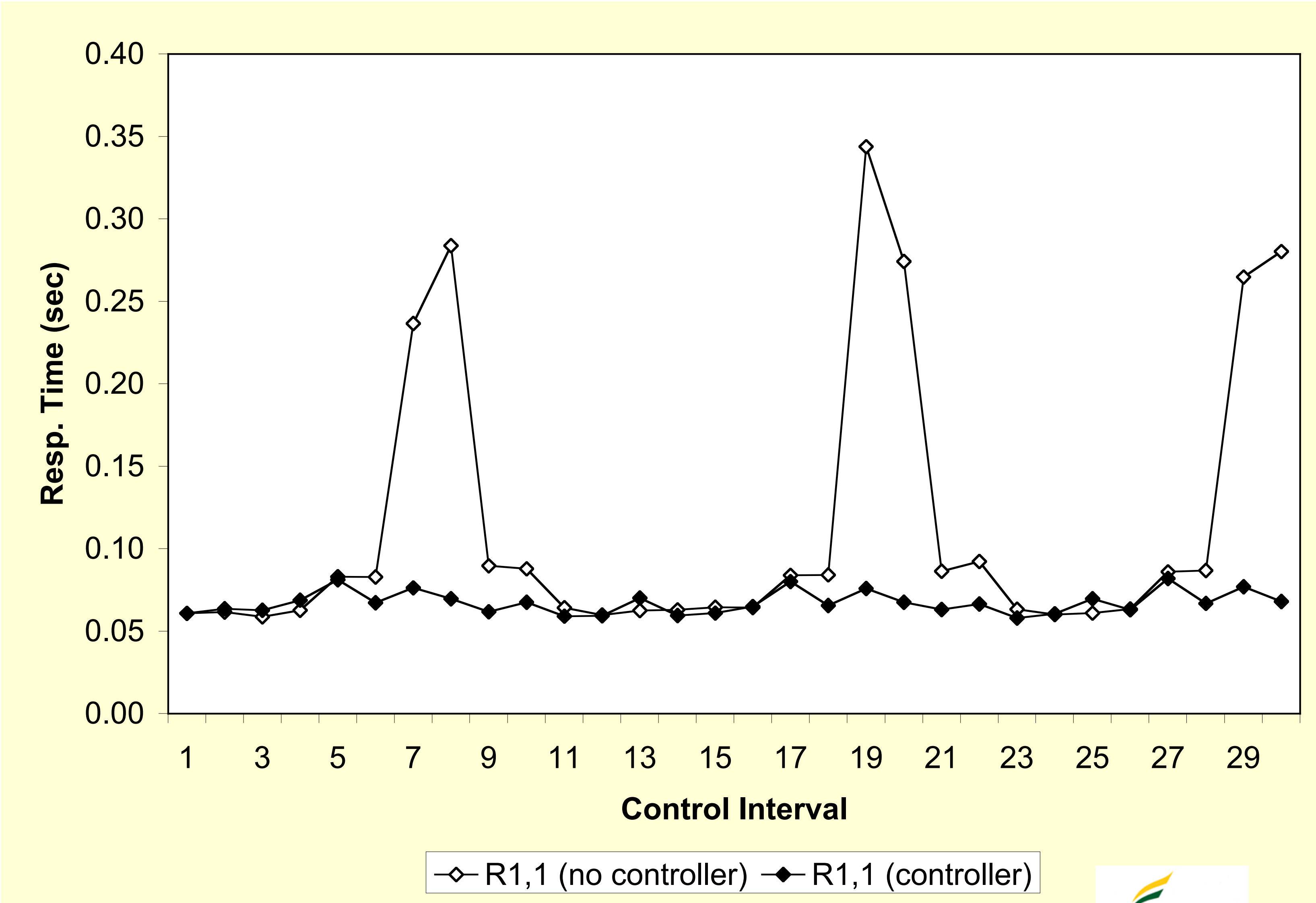
$$0 < a_{i,s} < 1$$

$$\sum_{s=1}^{S_i} a_{i,s} = 1$$

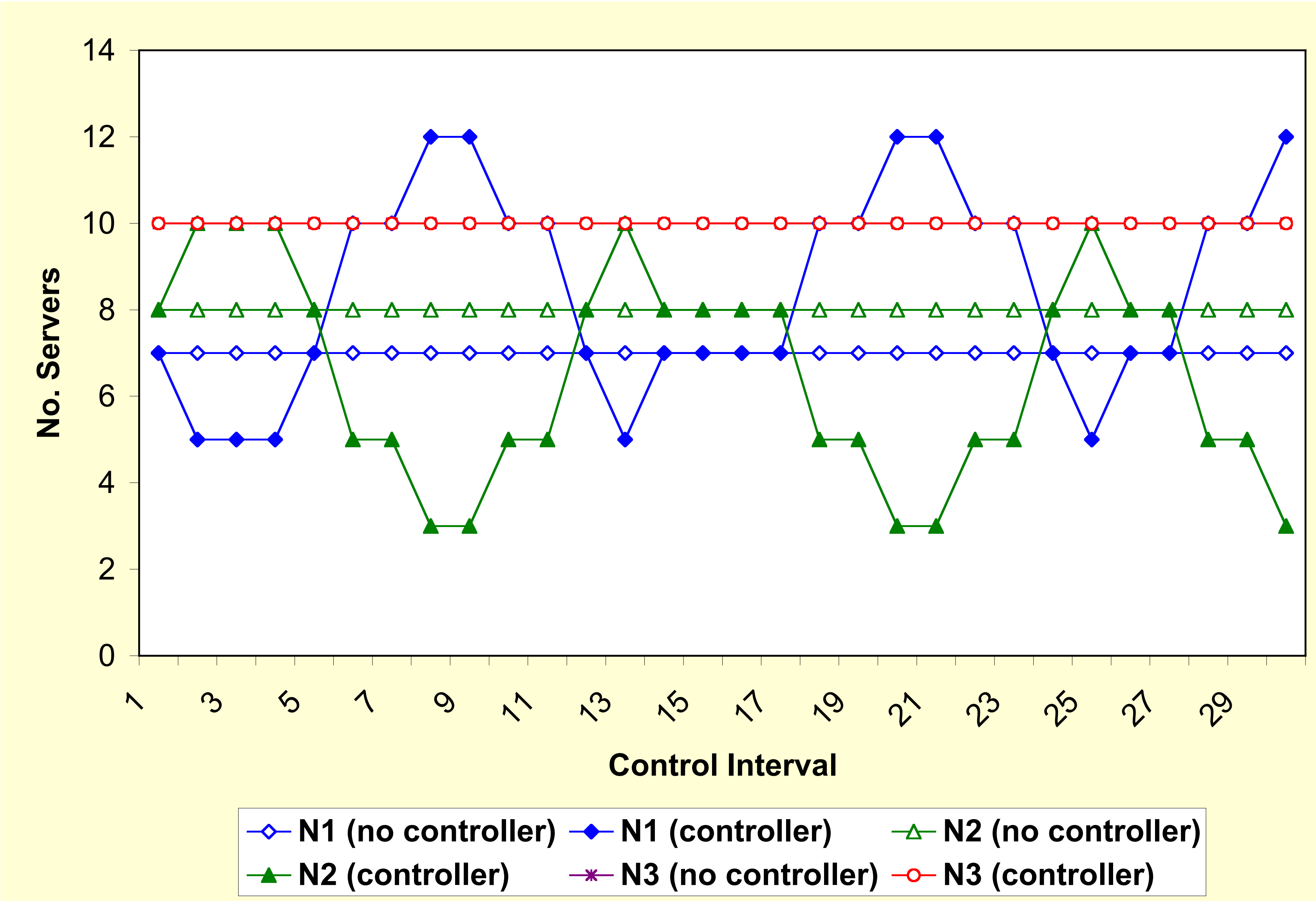
# Workload Variation for Online AEs



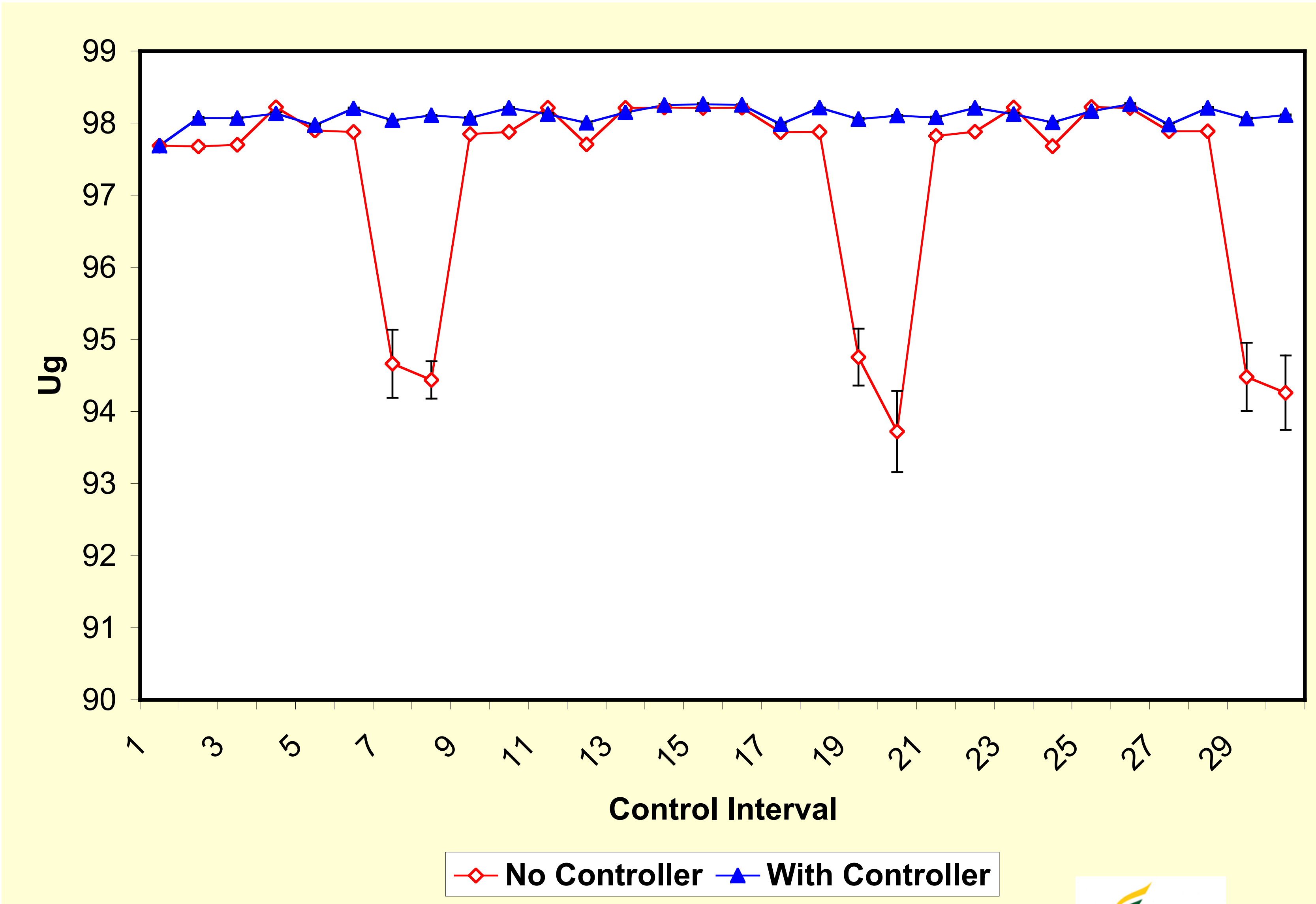
# Response Times for Class 1 of AE 1



# Variation of the Number of Servers



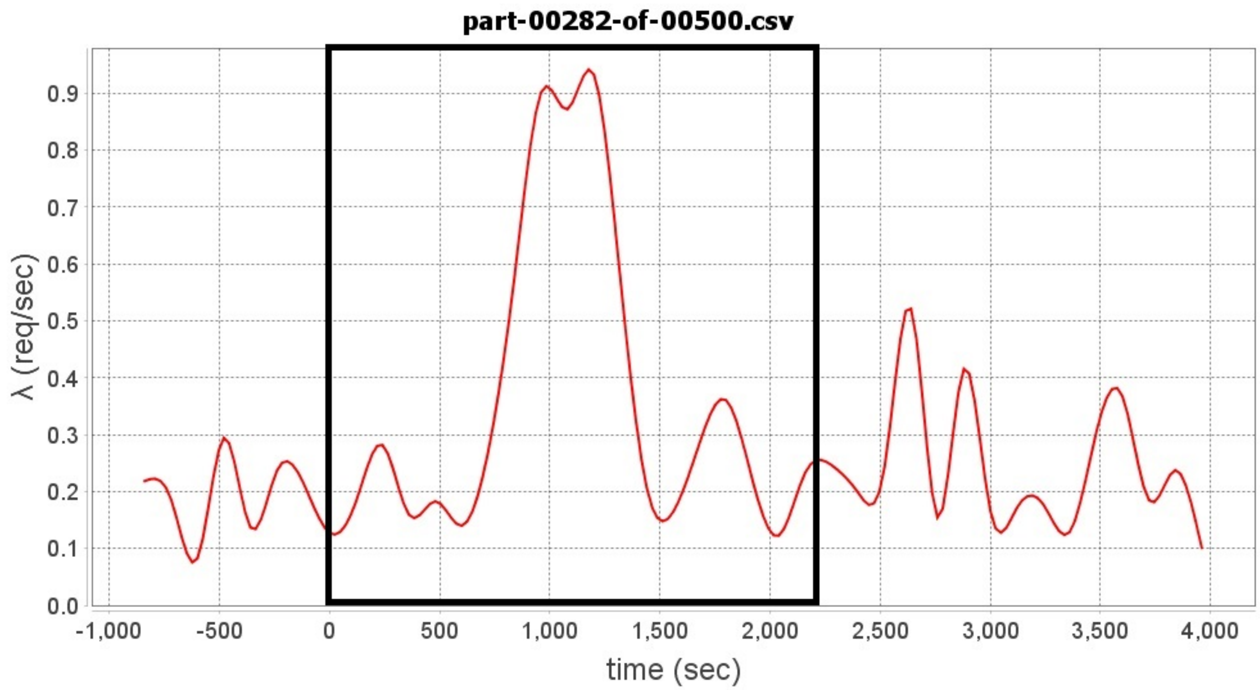
# Variation of Global Utility



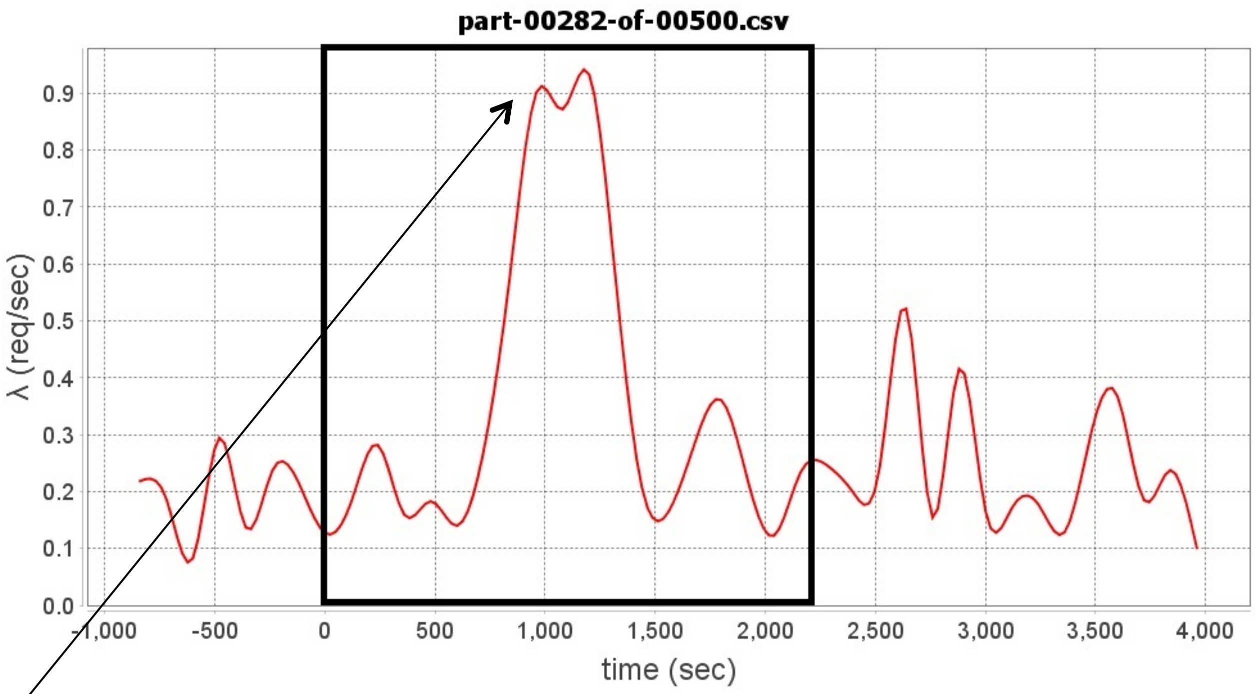
# TAMING WORKLOAD SURGES

- Model-Driven Elasticity Control for Multi-Server Queues Under Traffic Surges in Cloud Environments, V. Tadakamalla and D.A. Menasce, 2018 International Conf. on Autonomic Computing, Trento, Italy, September 3-7, 2018.
- An Analytic Model of Traffic Surges for Multi-Server Queues in Cloud Environments, V. Tadakamalla and D.A. Menasce, IEEE CLOUD 2018 Conf. July 2-7, 2018, San Francisco, CA, USA.
- Analysis and Autonomic Elasticity Control for Multi-Server Queues Under Traffic Surges, V. Tadakamalla and D.A. Menasce, *2017 IEEE Intl. Conf. Cloud and Autonomic Computing (ICCAC)*, Tucson, AZ, USA, September 18-22, 2017.

From Google Trace

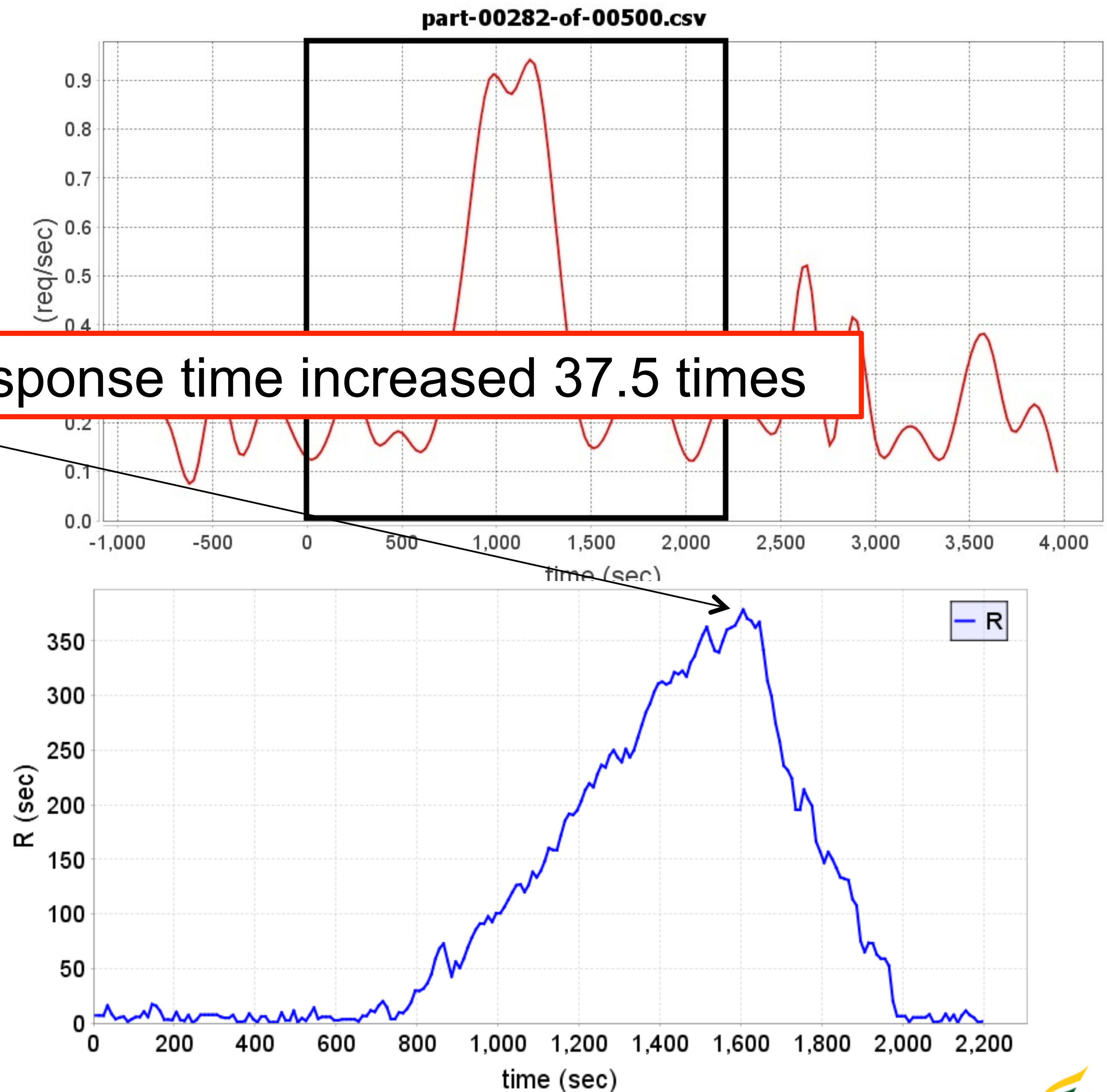


From Google Trace

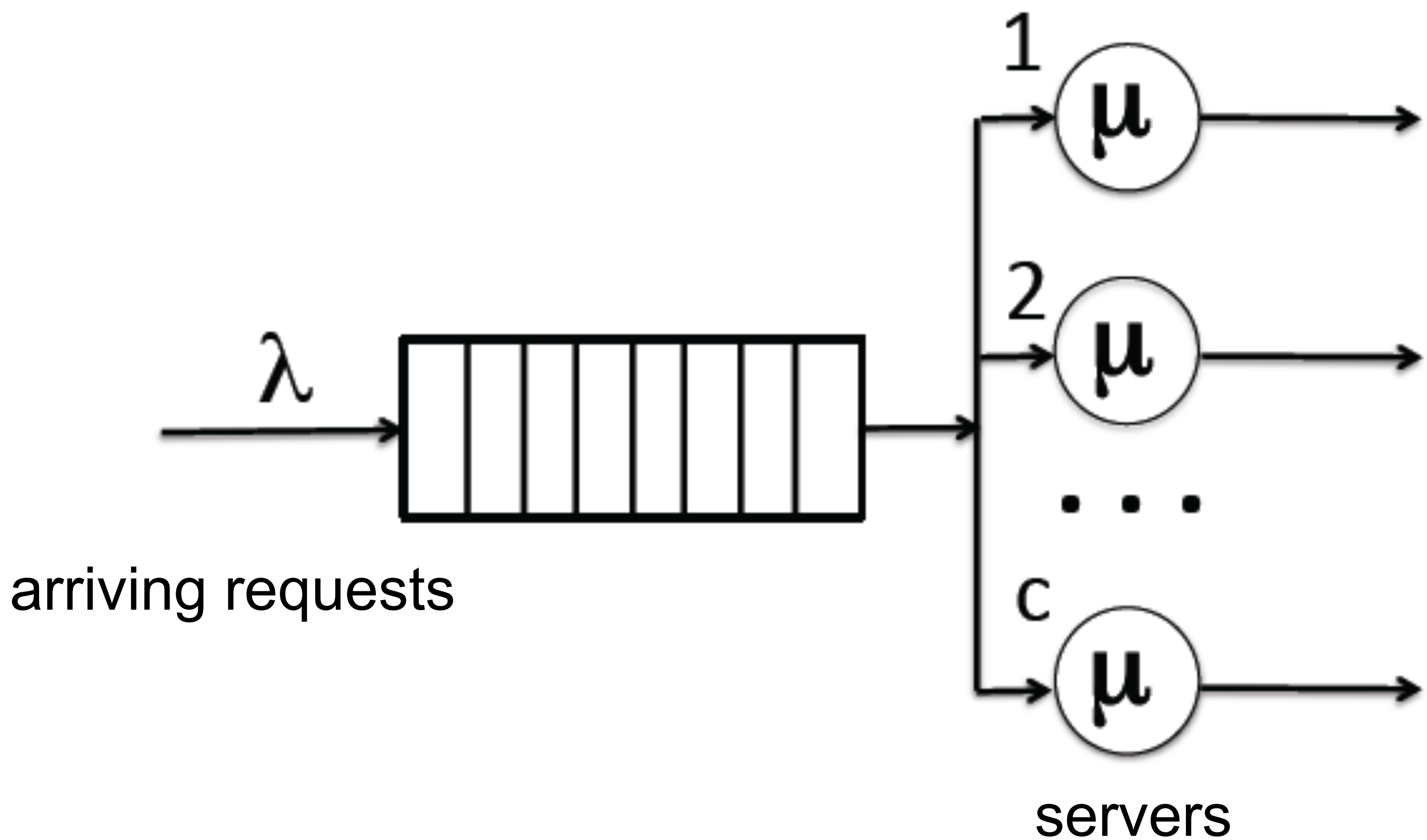



Workload increased by a 4.5 factor

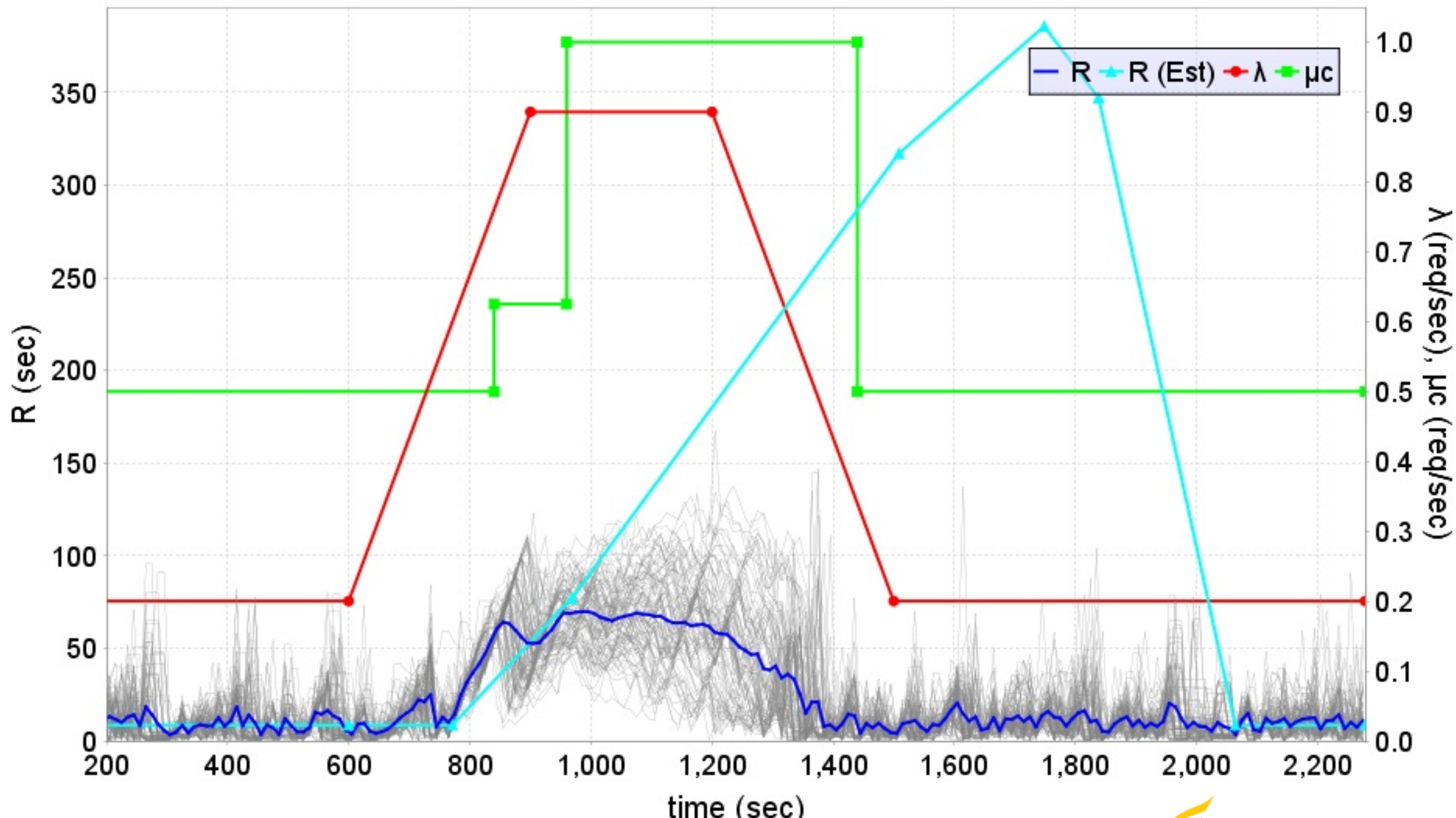
Peak response time increased 37.5 times



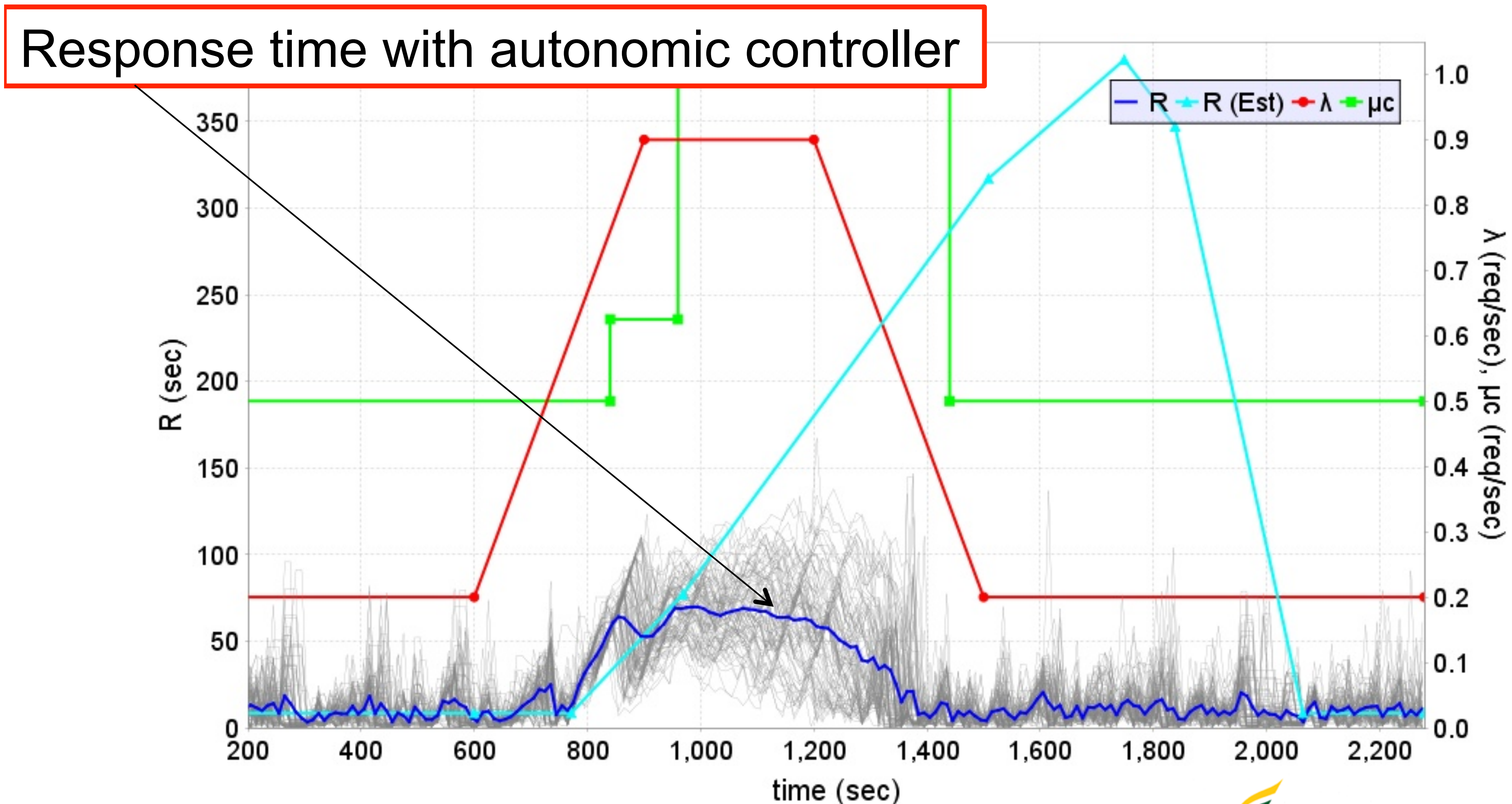
# Cloud Elasticity Control



# Cloud Elasticity Control



# Cloud Elasticity Control



# AUTONOMIC ENERGY-PERFORMANCE CONTROL

- Modeling the Tradeoffs Between System Performance and CPU Power Consumption, D.A. Menasce, 2015 Computer Measurement Group Conf., November 2-5, 2015, San Antonio, Texas.

# Energy Consumption

- Some estimates about Google:
  - One search: turn on a 60W bulb for 17 seconds
  - Google datacenters collectively burn 260 million Watts (1/4 output of a nuclear power plant)
  - Enough energy to continuously power 200,000 homes

# Modern CPUs

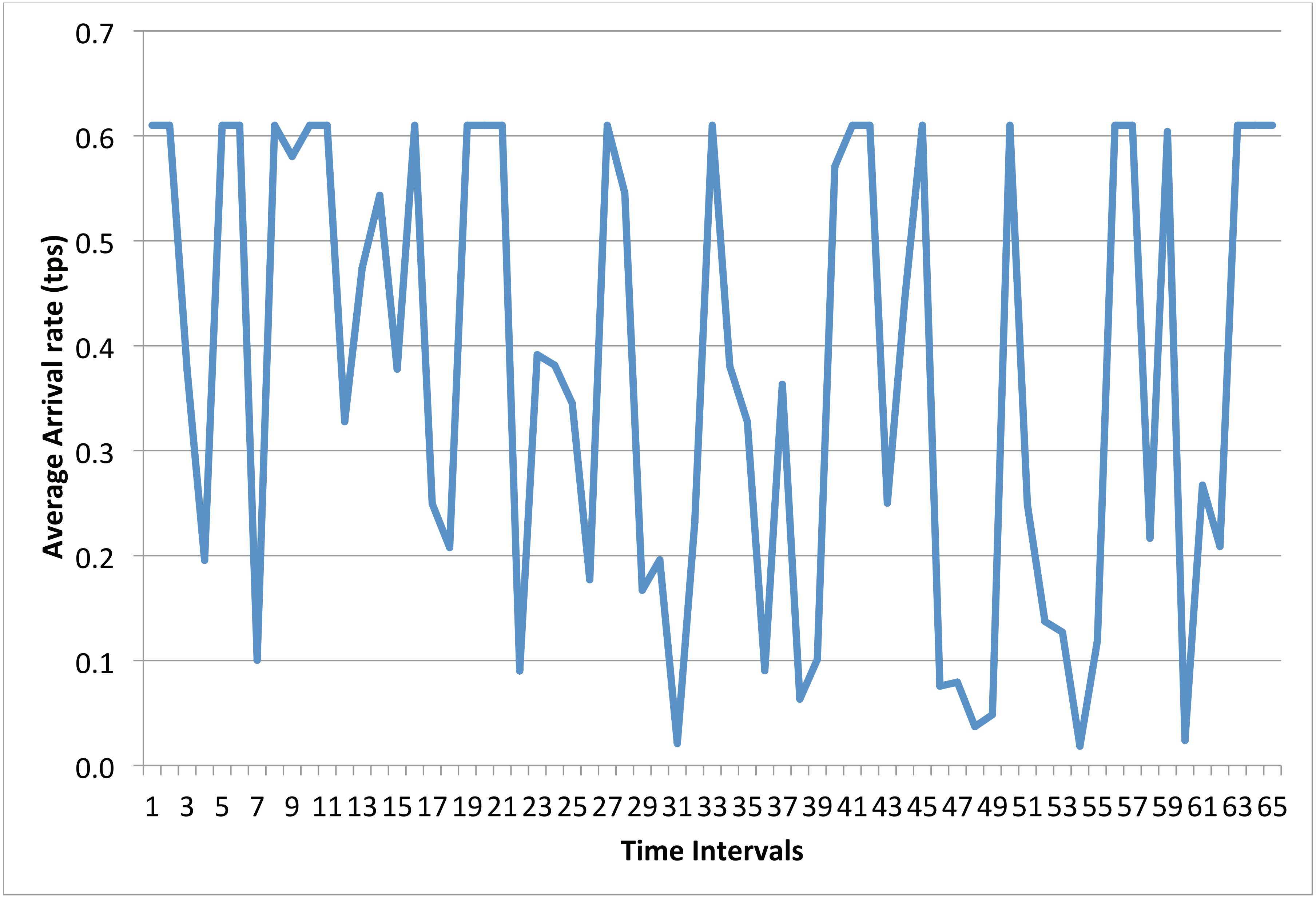
- Dynamic Voltage and Frequency Scaling (DVFS)
- Dynamic power is proportional to

$$V^2 \times f$$

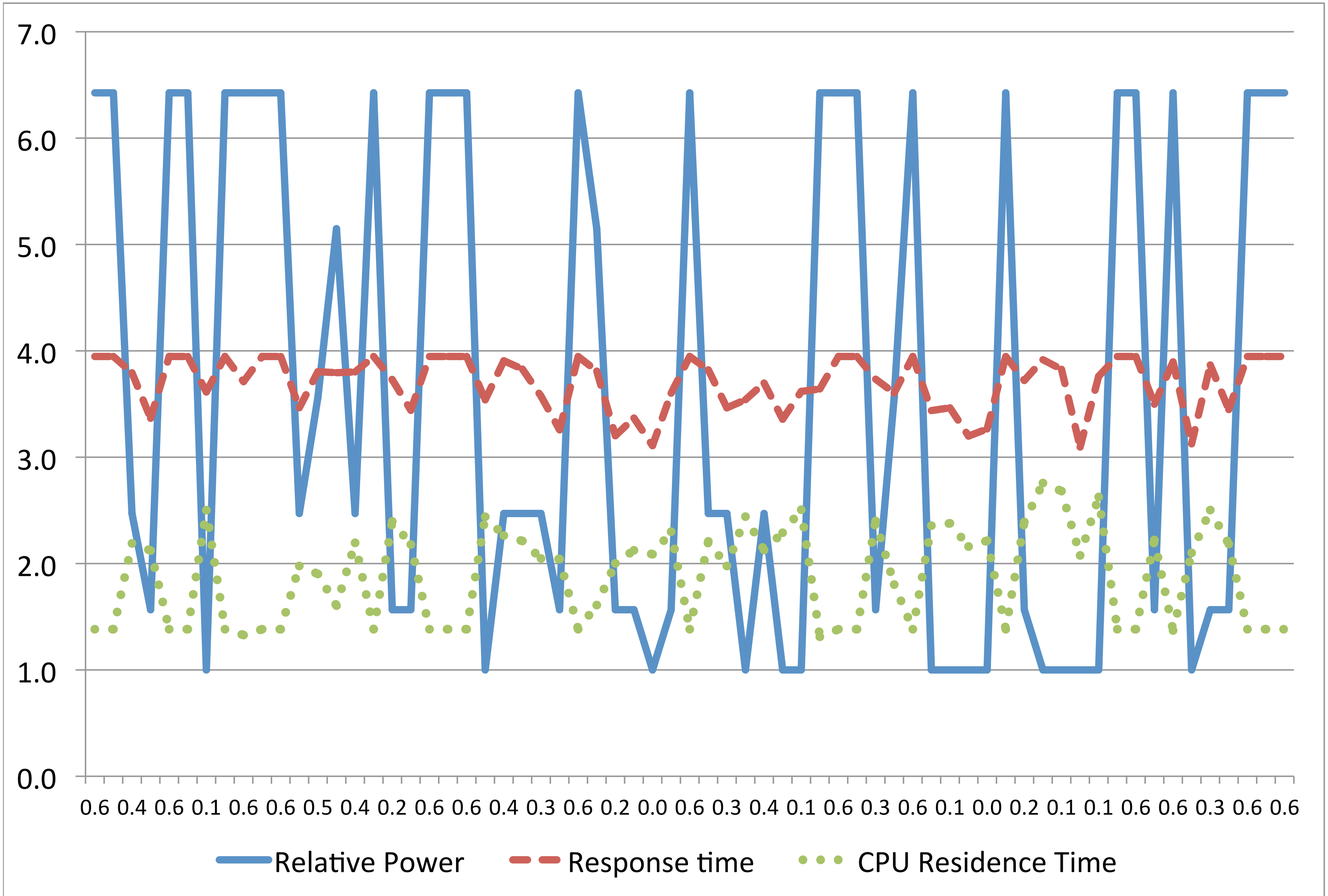
CPU voltage — CPU clock rate

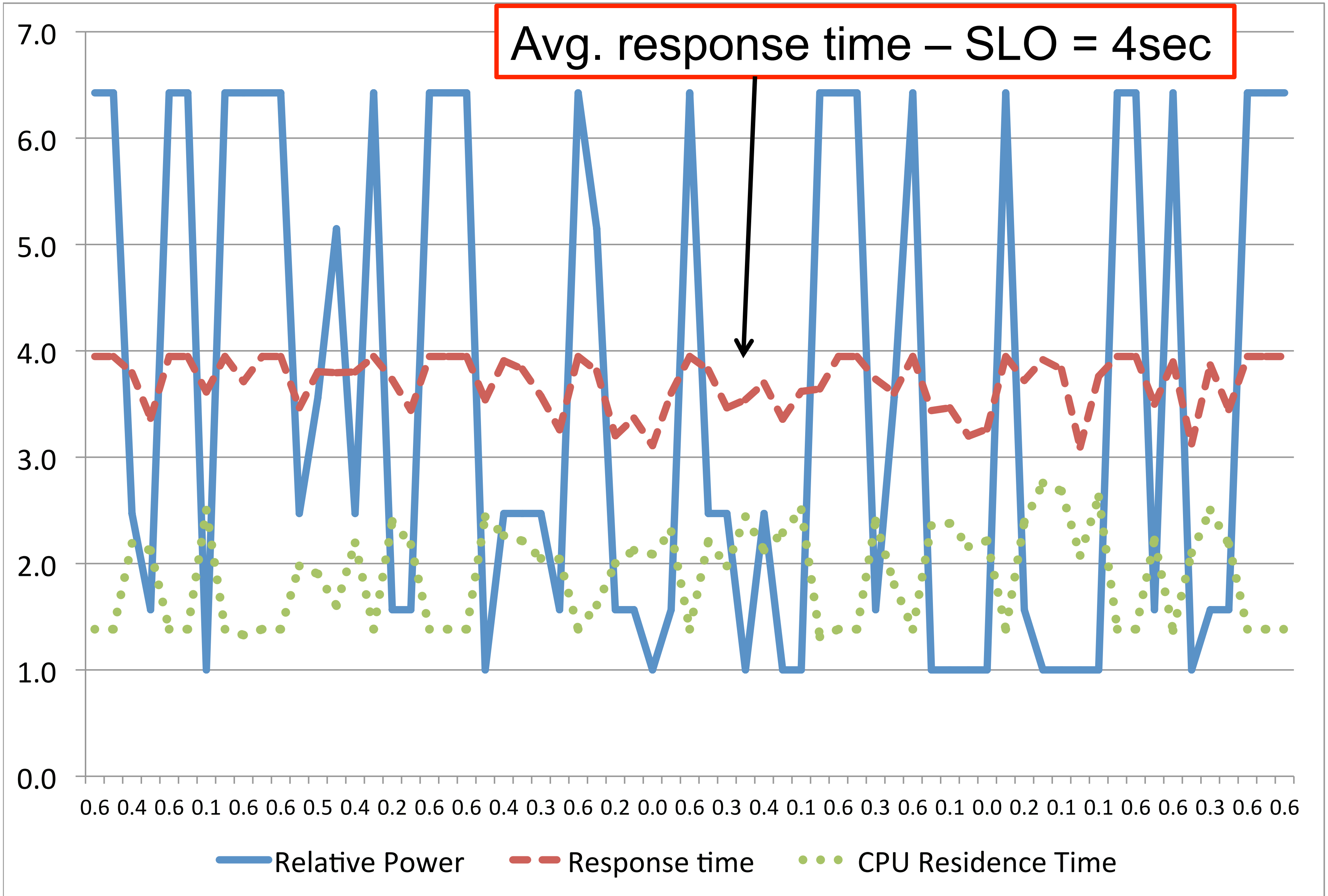
Need to dynamically vary the voltage-frequency pair in order to minimize energy consumption while meeting performance goals.

# Workload Variation Over Time

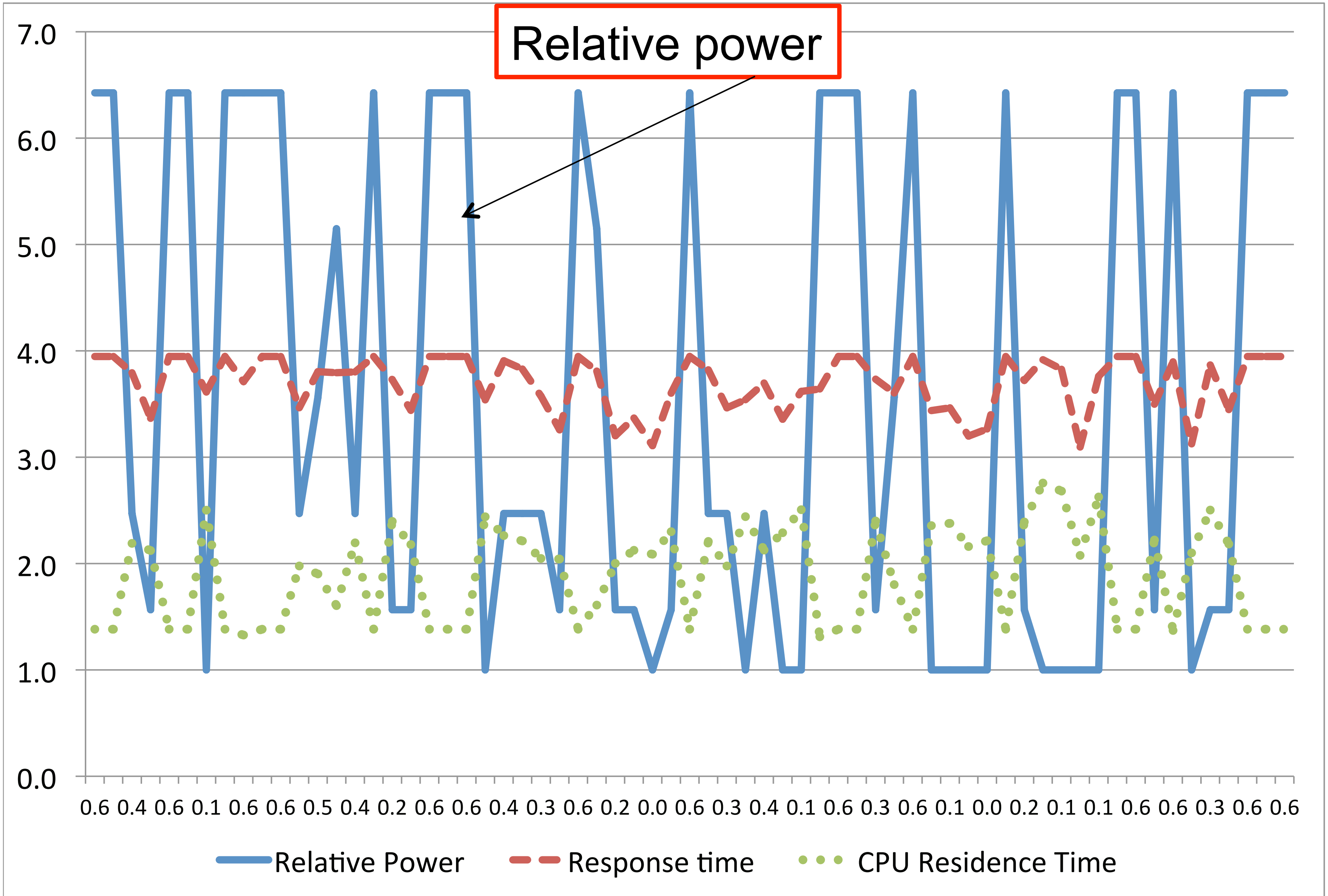


# Autonomic Power Variation





Avg. response time – SLO = 4sec
7.0
6.0
5.0
4.0
3.0
2.0
1.0
0.0
0.6 0.4 0.6 0.1 0.6 0.6 0.5 0.4 0.2 0.6 0.6 0.4 0.3 0.6 0.2 0.0 0.6 0.3 0.4 0.1 0.6 0.3 0.6 0.1 0.0 0.2 0.1 0.1 0.6 0.6 0.3 0.6 0.6
Relative Power
Response time
CPU Residence Time



Relative power
7.0
6.0
5.0
4.0
3.0
2.0
1.0
0.0
0.6 0.4 0.6 0.1 0.6 0.6 0.5 0.4 0.2 0.6 0.6 0.4 0.3 0.6 0.2 0.0 0.6 0.3 0.4 0.1 0.6 0.3 0.6 0.1 0.0 0.2 0.1 0.1 0.6 0.6 0.3 0.6 0.6
Relative Power
Response time
CPU Residence Time

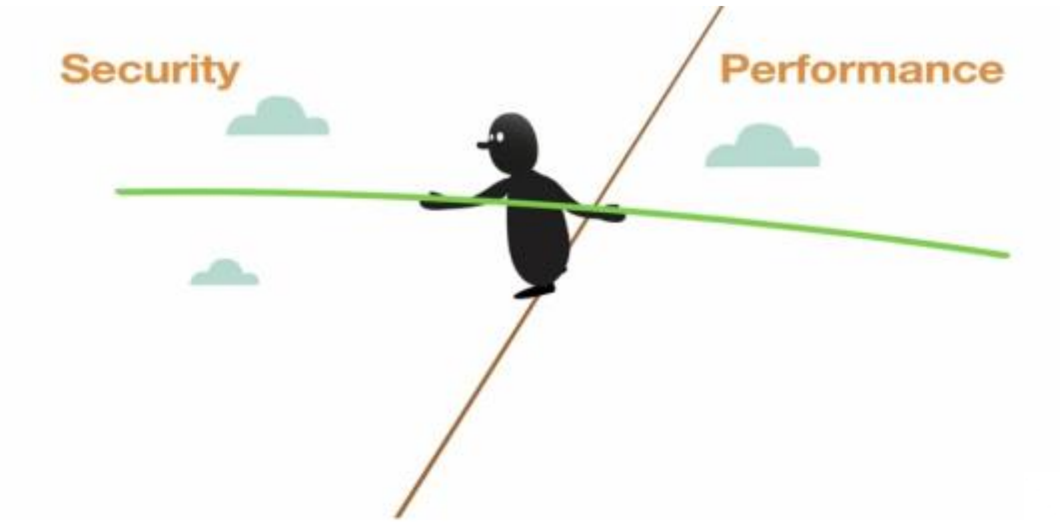
# DATABASE SECURITY-PERFORMANCE CONTROL

- Self-Protecting and Self-Optimizing Database Systems: Implementation and Experimental Evaluation, F. Alomari and D.A. Menasce, *The ACM Cloud and Autonomic Computing Conference (CAC 2013),* Miami, FL, August 5-9, 2013.

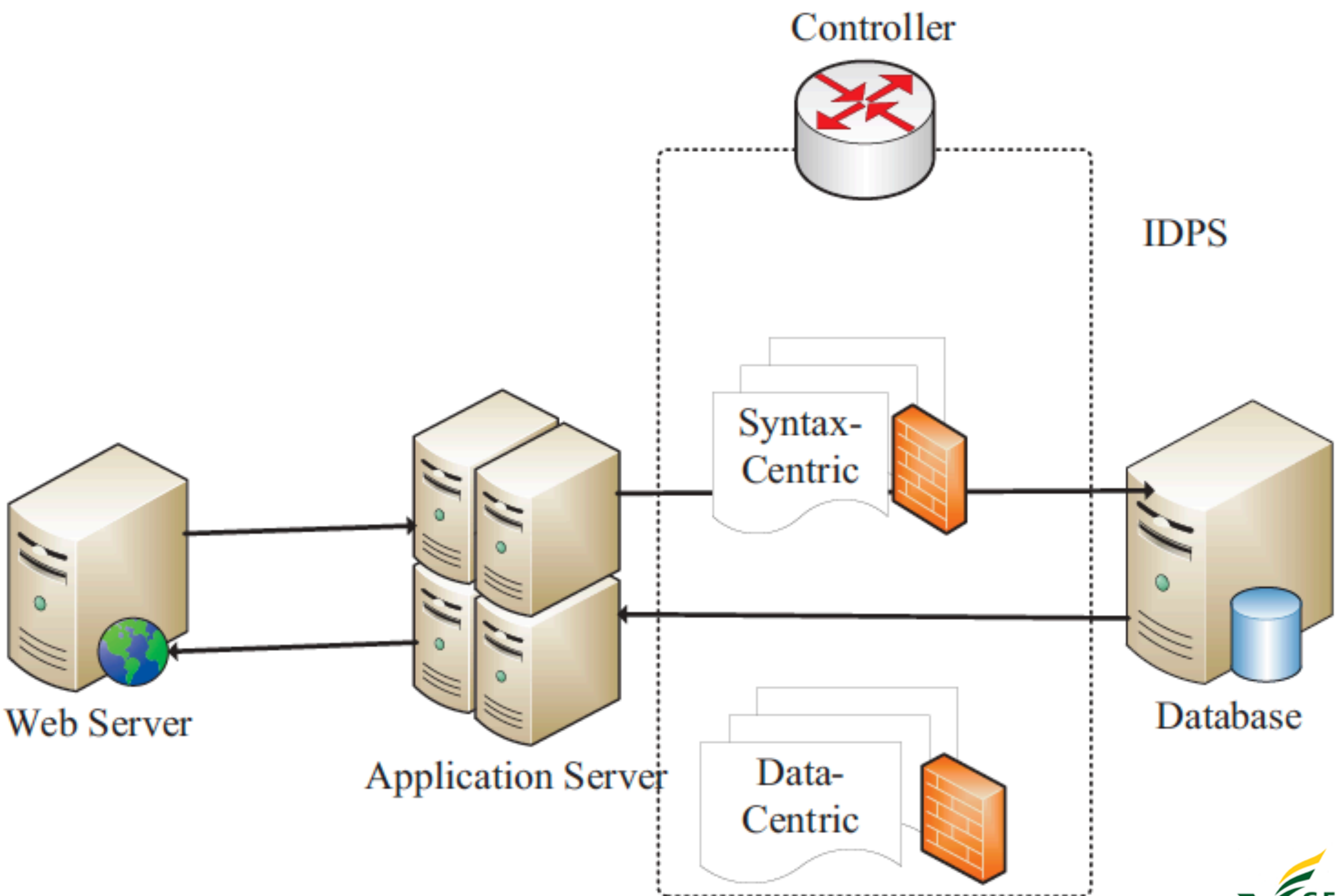
- An Autonomic Framework for Integrating Security and Quality of Service Support in Databases, F. Alomari and D.A. Menasce, *IEEE Sixth International Conference on Software Security and Reliability (SERE 2012),* , June 20-22, 2012, Washington, D.C., USA.

# Security and Performance Tradeoffs

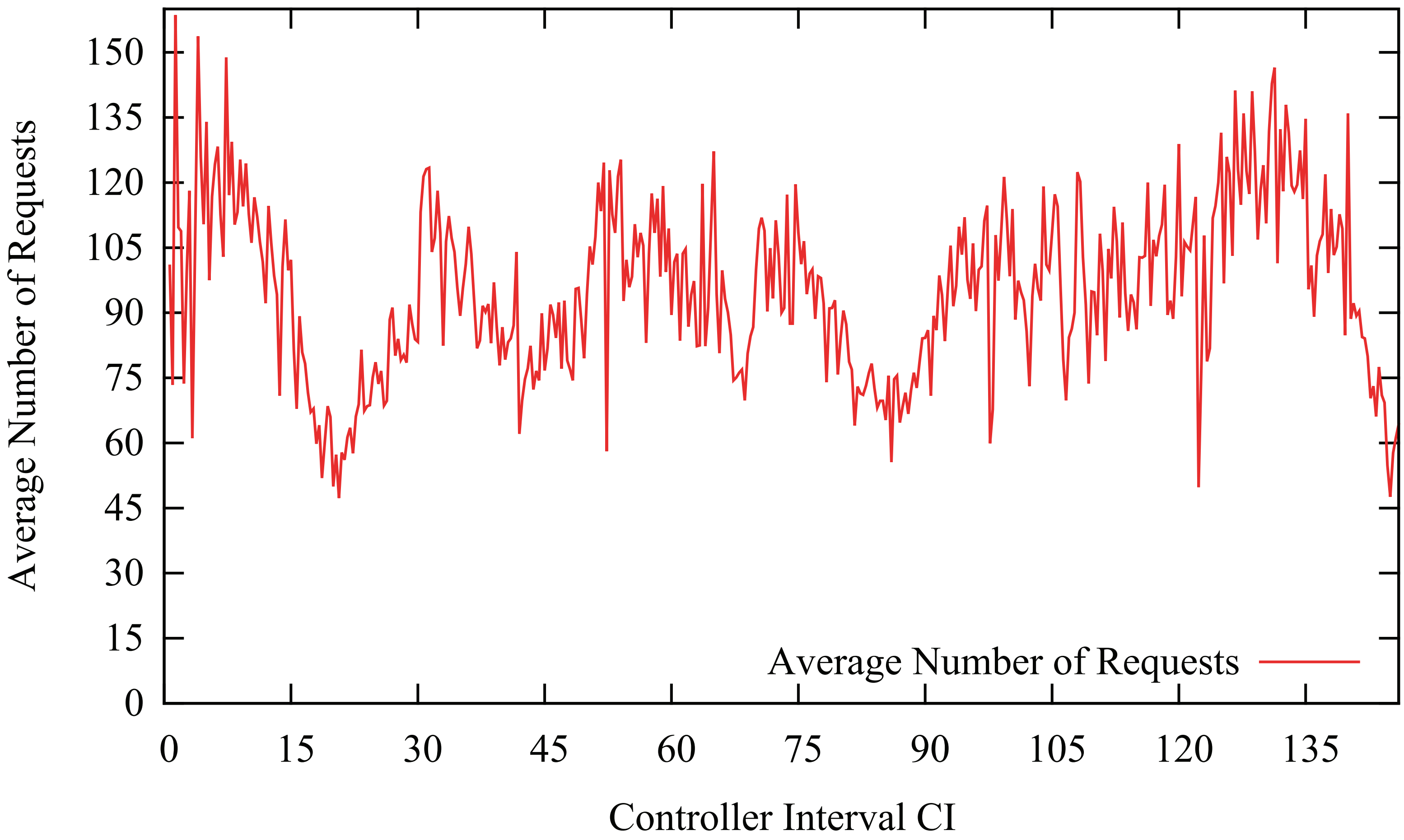
- Changes in security and performance objectives are no longer occasional occurrences, but expected events, and need to be dealt with dynamically at run time.
- Need a proper balance between security and performance
  - Manual configuration  is very difficult and error prone.
  - Systems have to manage themselves to be practical and profitable.



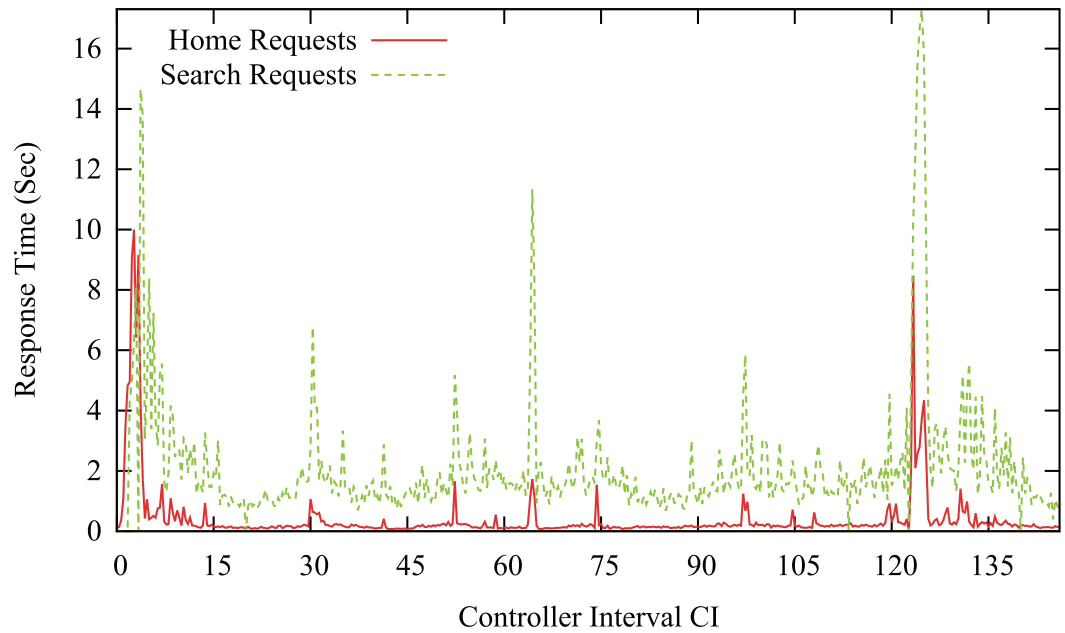
# Intrusion Detection Prevention Systems



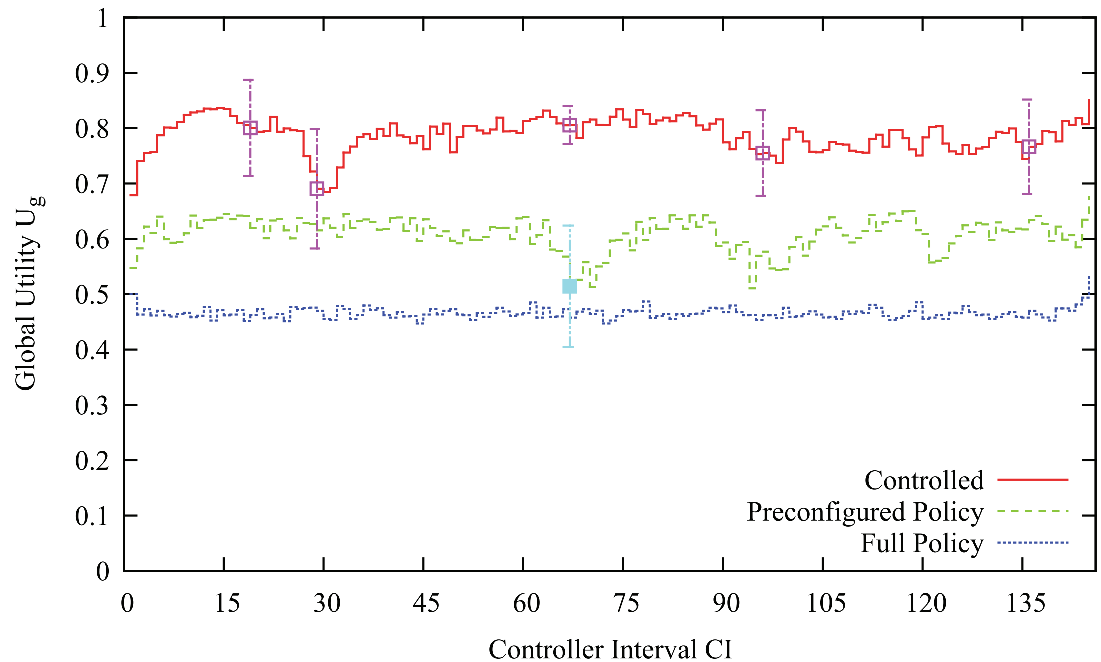
# Workload Intensity Over Time



# Response Time




# Utility



# Concluding Remarks

- Most complex systems are dynamic and evolve fast over time
- Not possible for human beings to constantly optimize them
- Self-managed systems automatically change configuration parameters based on high-level goals provided by human beings