Politecnico di Milano
Dip. Elettronica, Informazione e Bioingegneria
Milano, Italy

Performance Engineering

Learning through Applications using JMT

Open Access book – Springer - 2023

https://link.springer.com/book/10.1007/978-3-031-36763-2

Slides of case studies

Java Modelling Tools

Open Source Tools https://jmt.sourceforge.net

V2

Giuseppe Serazzi

26/11/2023

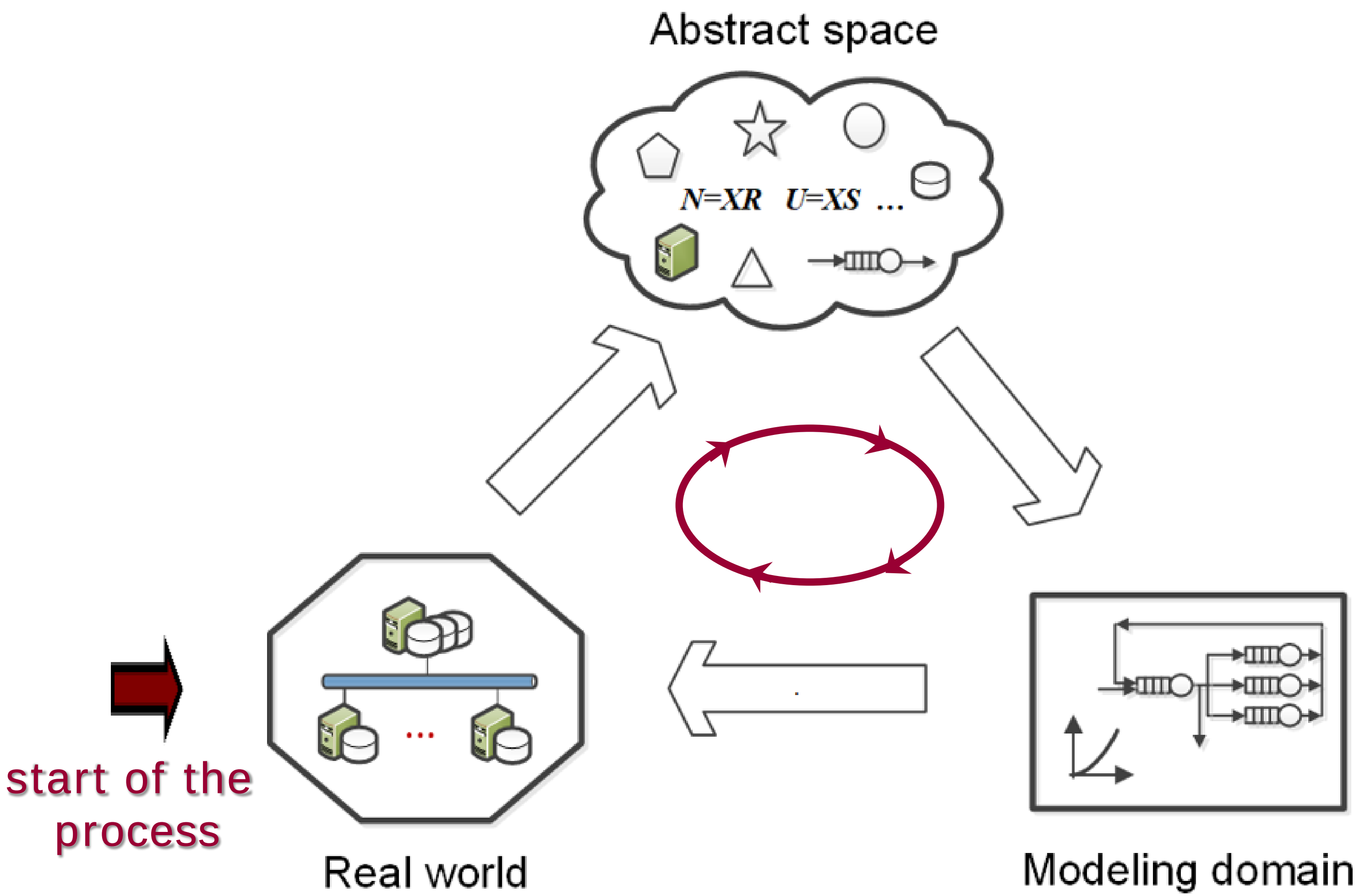
outline

- Chap.1 – Introduction to performance models
- Chap.2 – Sect.2.2 – A Computing Infrastructure with a Closed Workload
- Chap.2 – Sect.2.3 – Equivalent model with Service Demands
- Chap.2 – Sect.2.4 – Optimal operating point of a server
- Chap.3 – Sect.3.2 – Impact of Bottleneck migration
- Chap.3 – Sect.3.3 – Performance Optimization of a Data Center
- Chap.4 – Sect.4.2 – Variability of Interarrival Times: impact on performance
- Chap.4 – Sect.4.3 – Variability of Service Times: impact on performance
- Chap.5 – Sec.5.1, 5.2, 5.3 – Parallel Computing
- Chap.6 – Sect.6.1 – A Facial Recognition Surveillance System (edge computing)
- Chap.6 – Sect.6.2 – Autoscaling Load Fluctuations
- Chap.6 – Sect.6.3 – Simulation of the Workflow of a Web/App

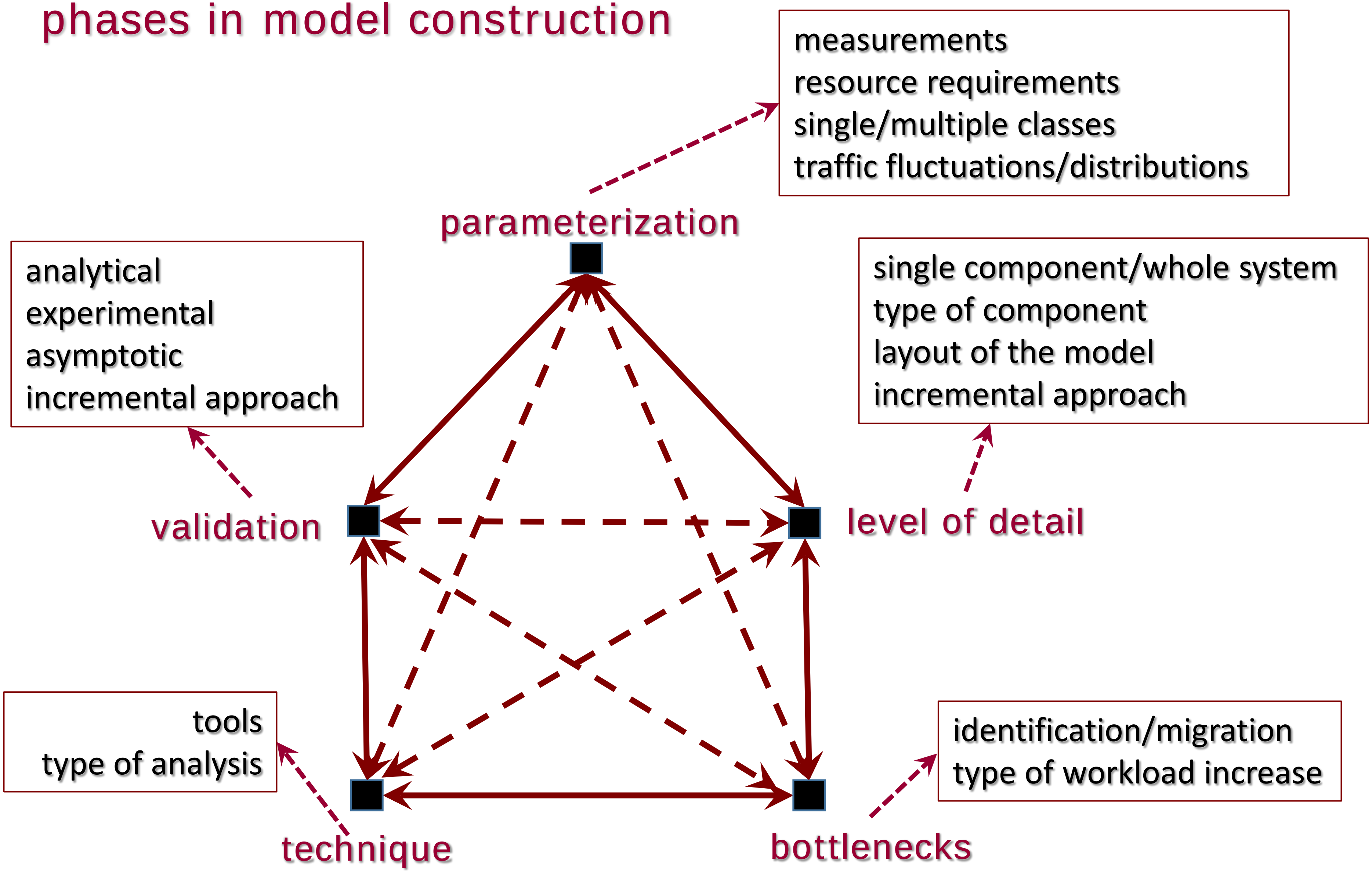
Introduction to Performance Models

Chapter 1

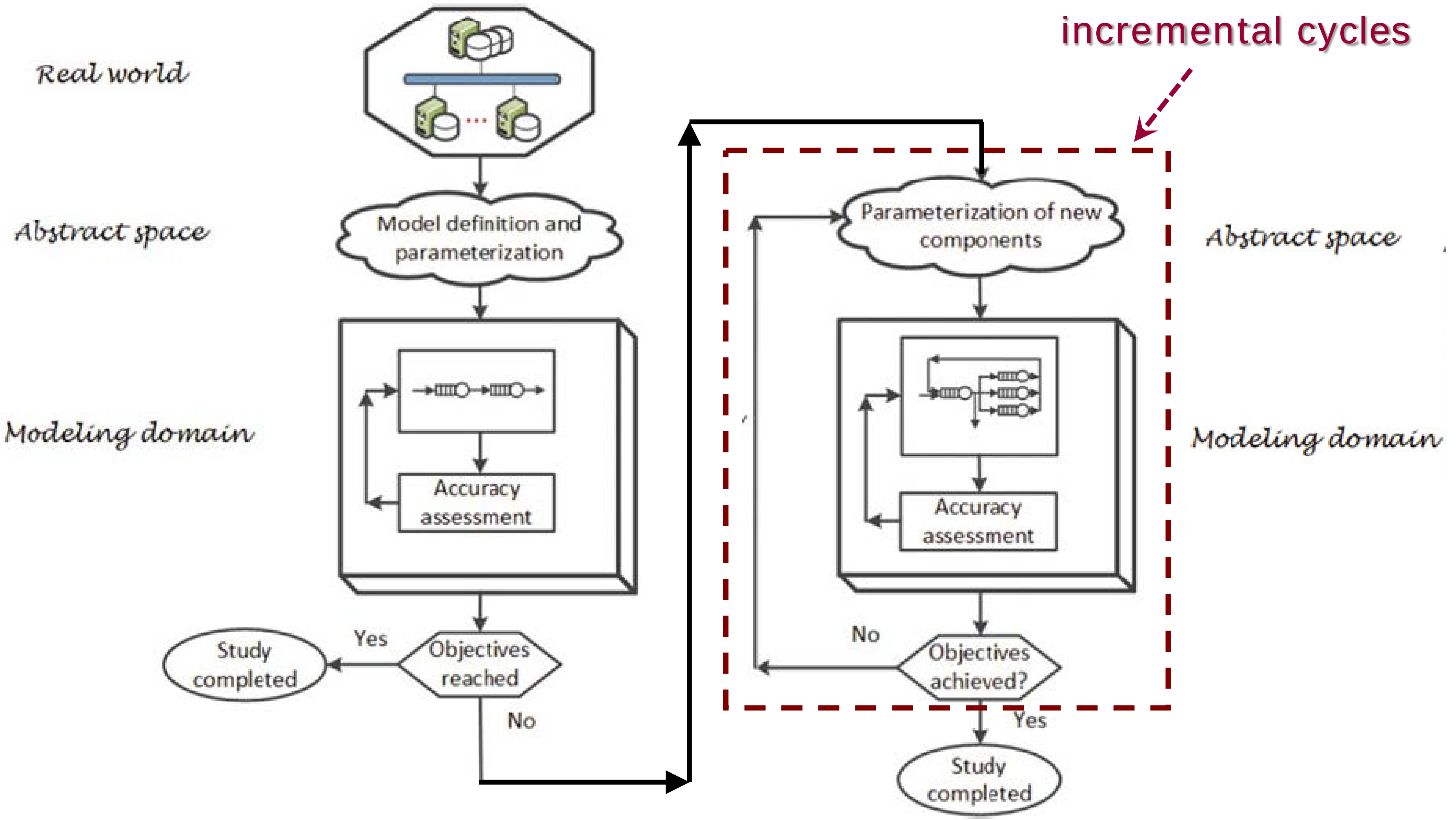
environments involved in modeling



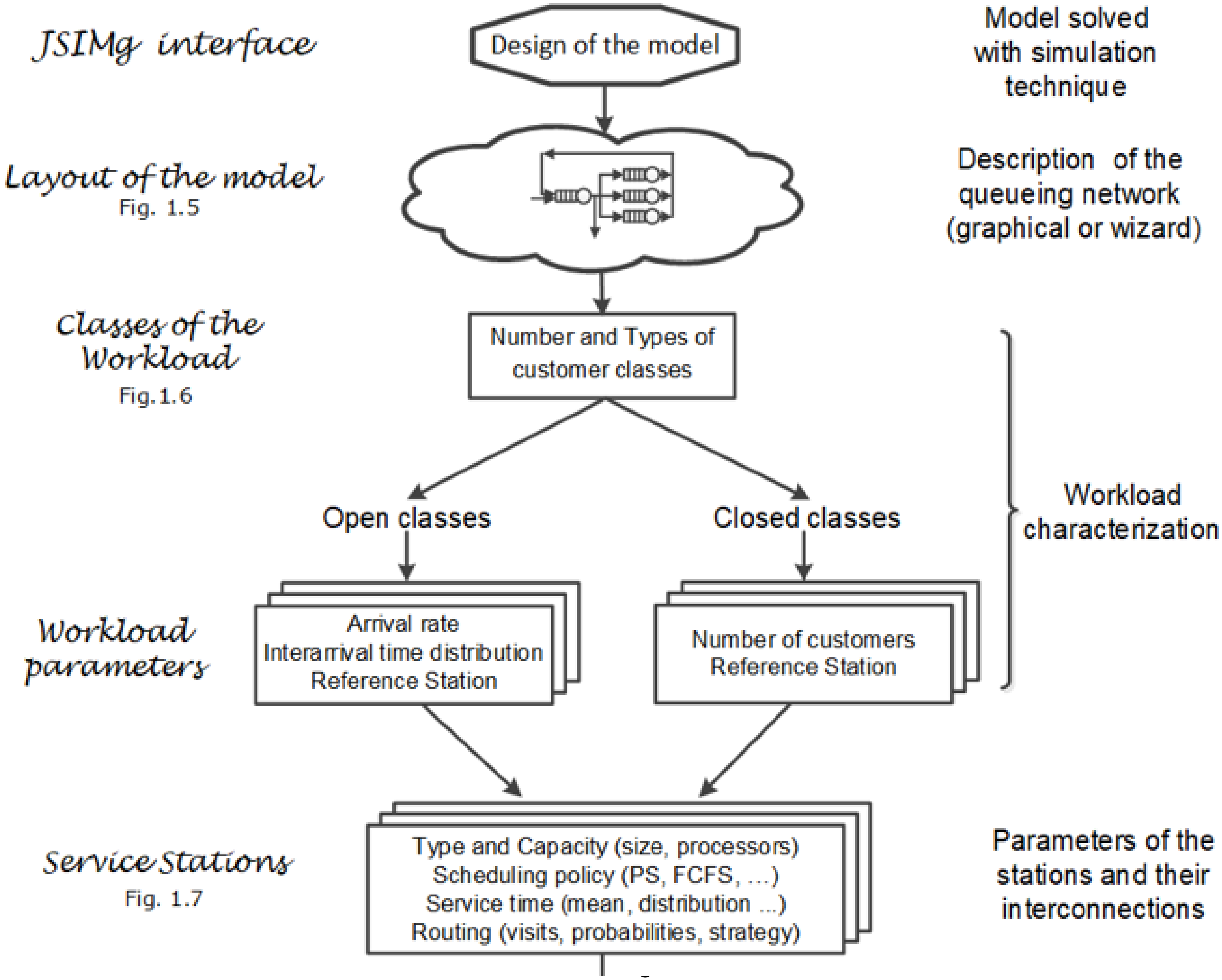
phases in model construction



incremental approach



implementation of a simulation model JSIMg (1)

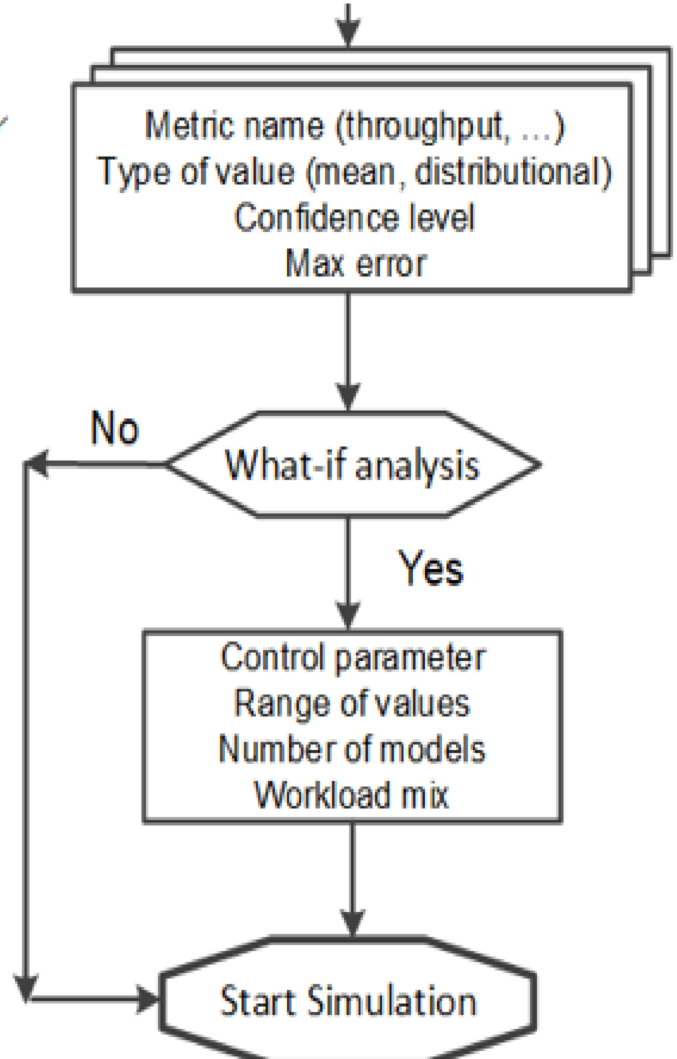


implementation of a simulation model JSIMg (2)

Performance metrics and results

Fig.s 1.8, 1.9

Description of the indexes to be collected



What-if analysis

Fig. 1.10

Execution of models with different parameters

A computing infrastructure with a closed workload

Chapter 2 --- Sect. 2.2

closed model
single class
tool used: JSIMg

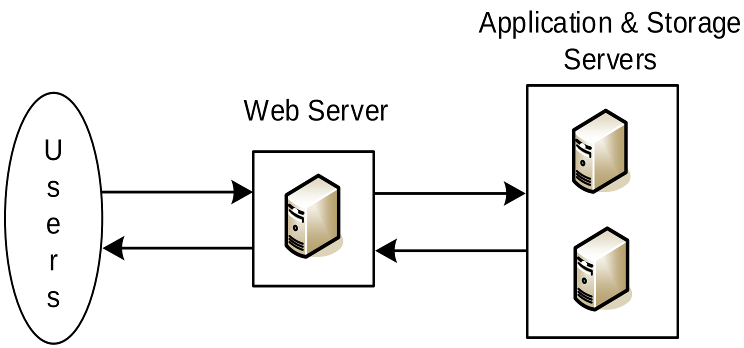
Sect.2.2 - the problem

- consider a data center with a limited (constant) number of users N_0
- the workload consists of one class of requests (that have similar resource requirements)
- three servers: a Web Server (WS), and two Application/Storage Servers (AS_1, AS_2)
- service times S_{WS}=0.005sec, S_{AS1}=0.020sec, S_{AS2}=0.025sec that are exponentially distributed,
- users think time Z=1sec

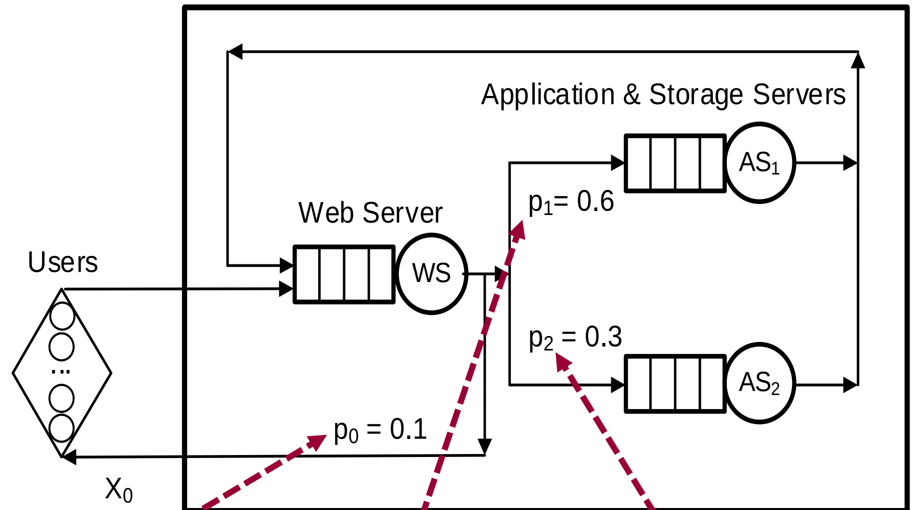
objectives

- study the behavior of X_0 and R_0 for $N_0=1÷20$ users (with the original configuration)
- compute the 90th percentile of R_0 for $N_0=20$
- evaluate the effects on performance of the upgrade of AS_2 (the slowest of the two App&Storage servers) with one 20% faster new S_{AS2} --> 0.020sec
- evaluate the effects on performance of the upgrade of AS_1 with on e 20% faster, --> $S_{AS1}=0.016$sec
- migration of bottleneck?
- forecast X_0 and R_0 with a workload of $N_0=40$ users (with the original configuration)

the computing infrastructure



(a)



(b)

Visits

$$V_{WS} = \frac{1}{p_0} = 10 \qquad V_{AS_1} = \frac{p_1}{p_0} = 6 \qquad V_{AS_2} = \frac{p_2}{p_0} = 3$$

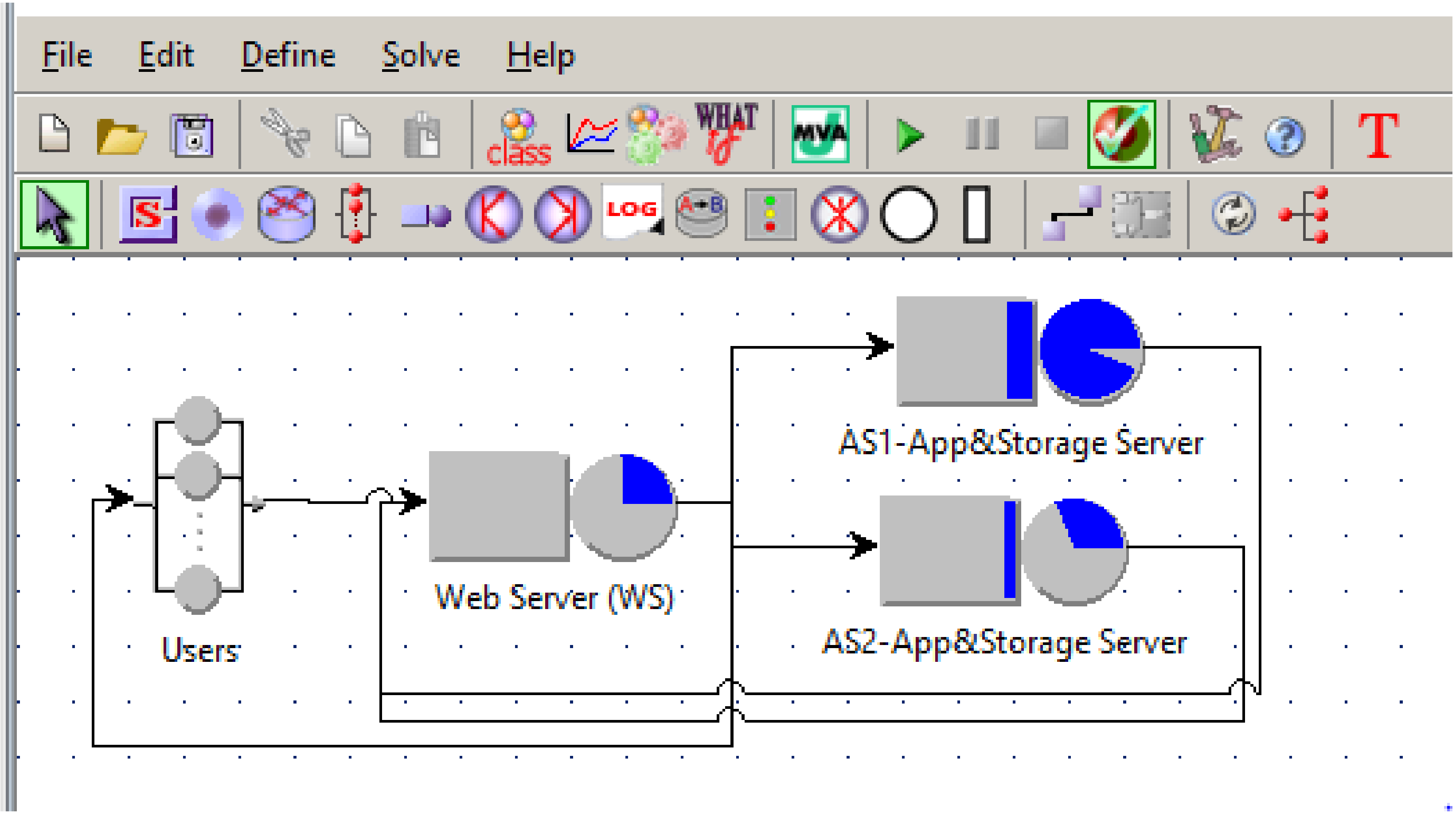
from routing probabilities to visits (see Sect. A1)

Service demands: D_r

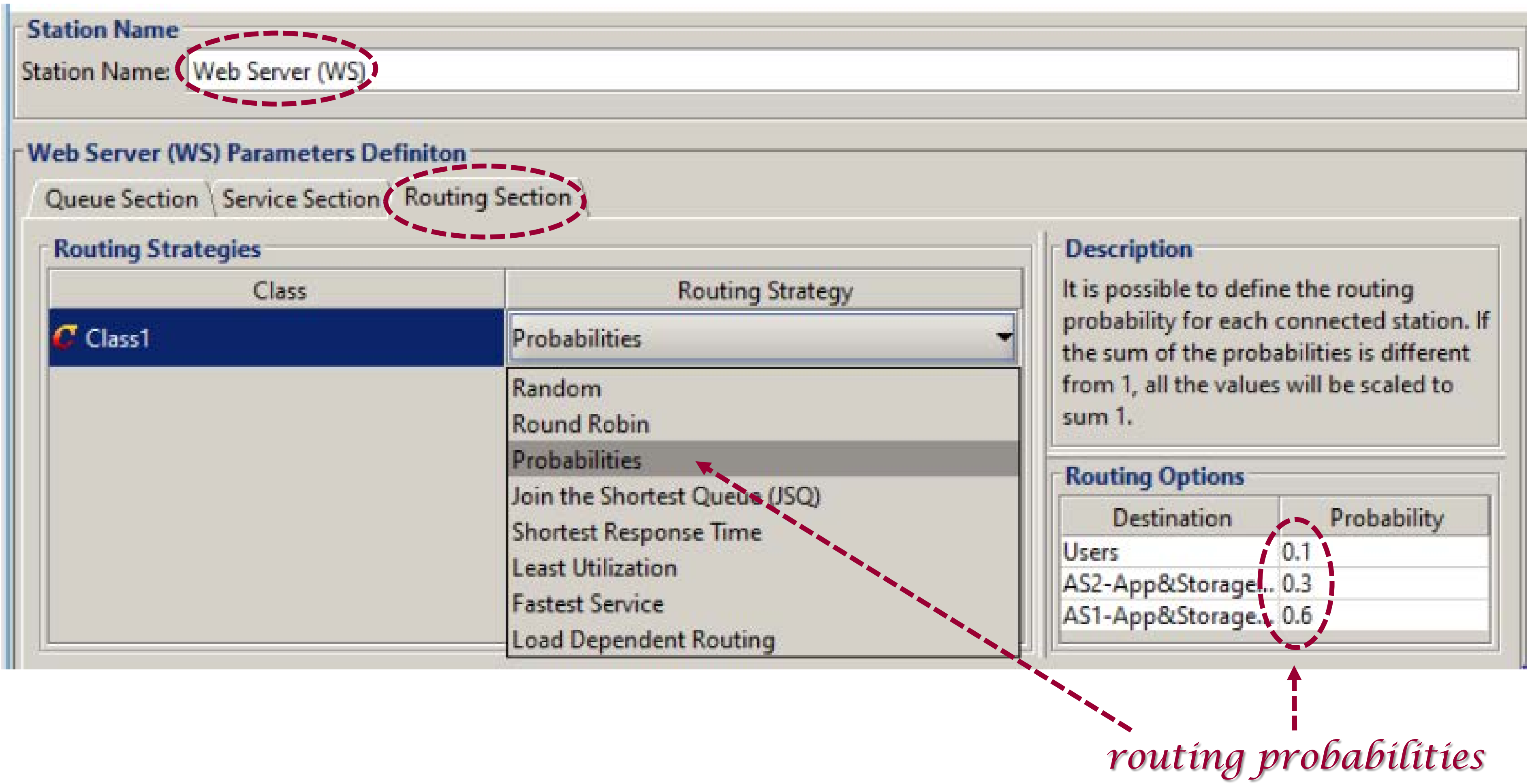
- $D_{WS} = V_{WS}\, S_{WS} = 0.05$ sec
- $D_{AS1} = V_{AS1}\, S_{AS1} = 0.12$ sec
- $D_{AS2} = V_{AS2}\, S_{AS2} = 0.075$ sec

- the **bottleneck** is on AS_1, max of D_i, despite that it is faster than AS2!

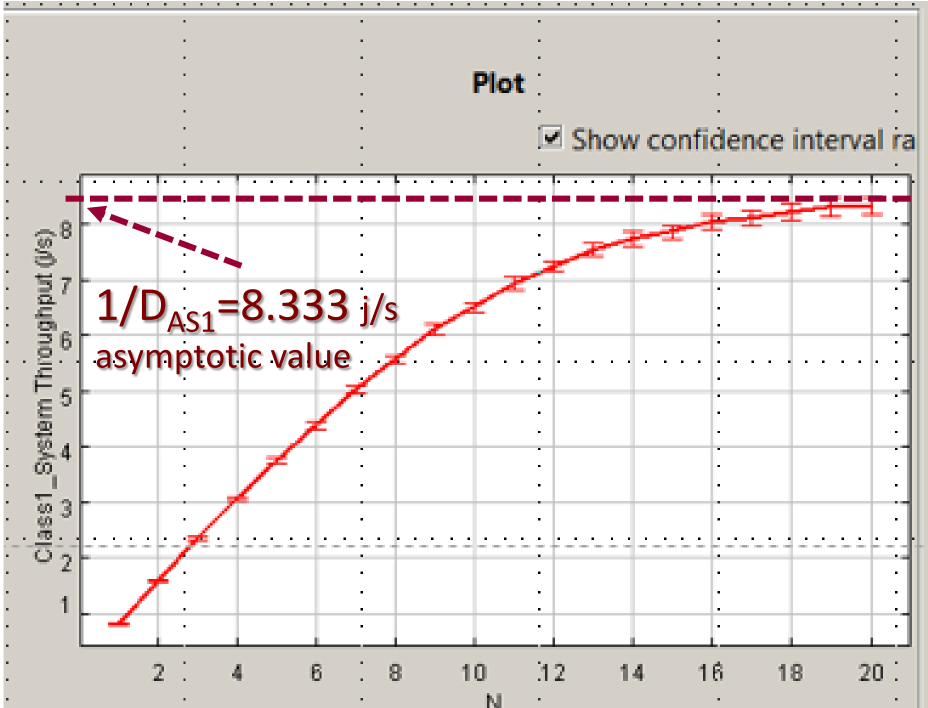
the JSIMg model



routing probabilities settings



Throughput and Response time of the original configuration

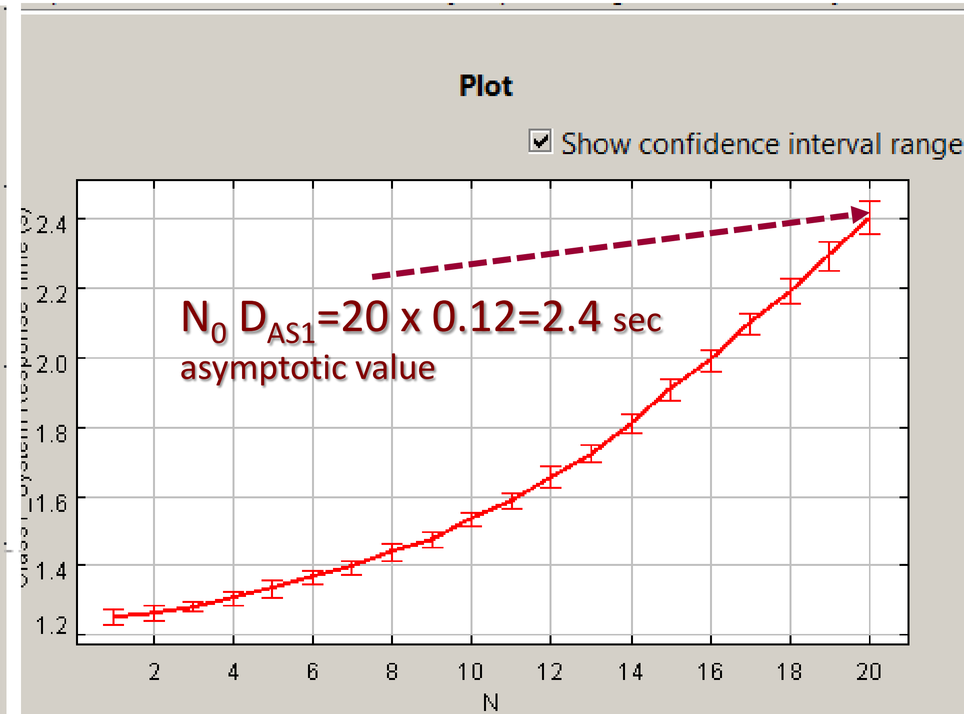


number of customers

Simulation Results

N = 16	N = 17	N = 18	N = 19	N = 20
8.039	8.105	8.214	8.305	8.321
8.183	8.239	8.375	8.484	8.470
7.900	7.976	8.058	8.134	8.177

System Throughput X_o

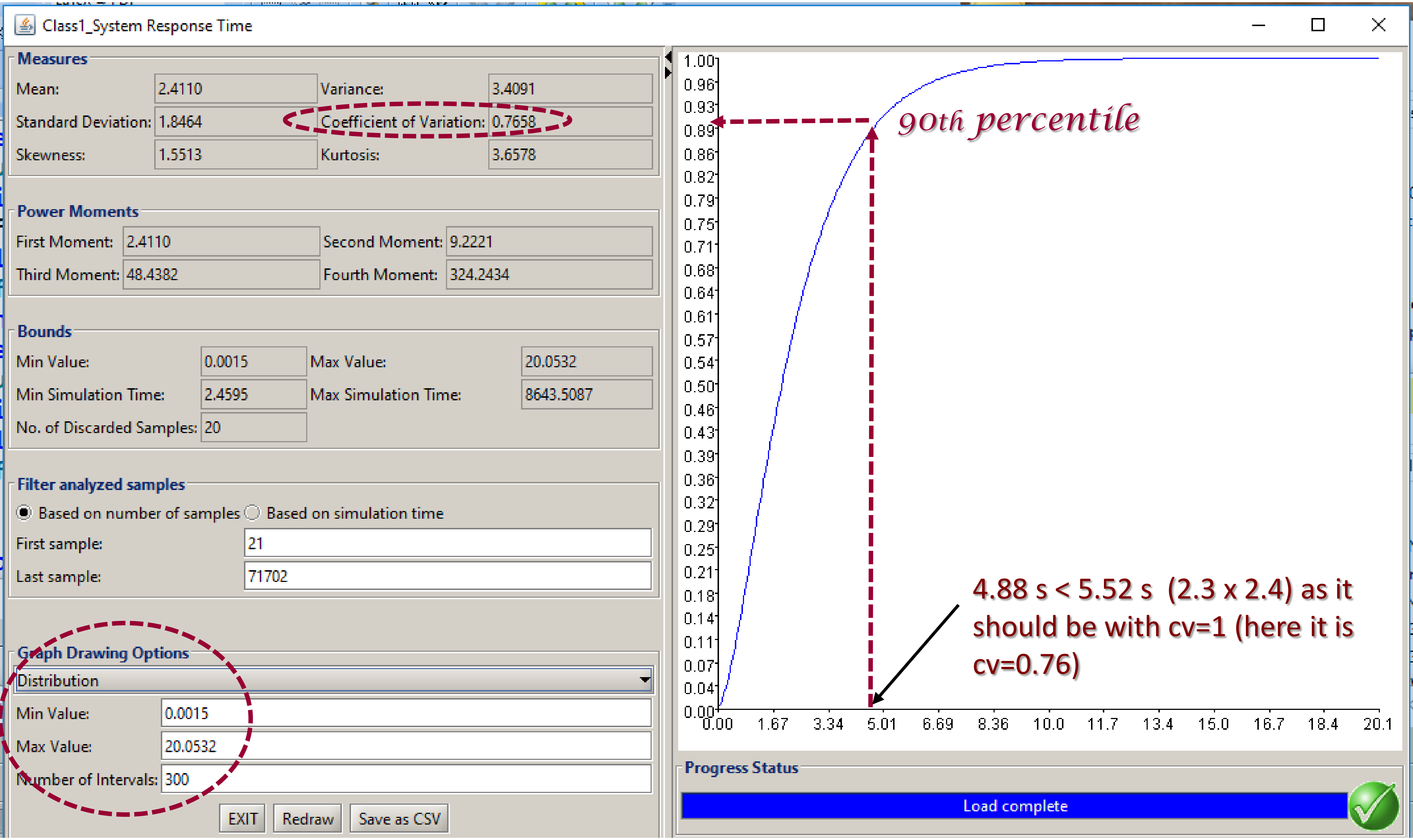


Simulation Results

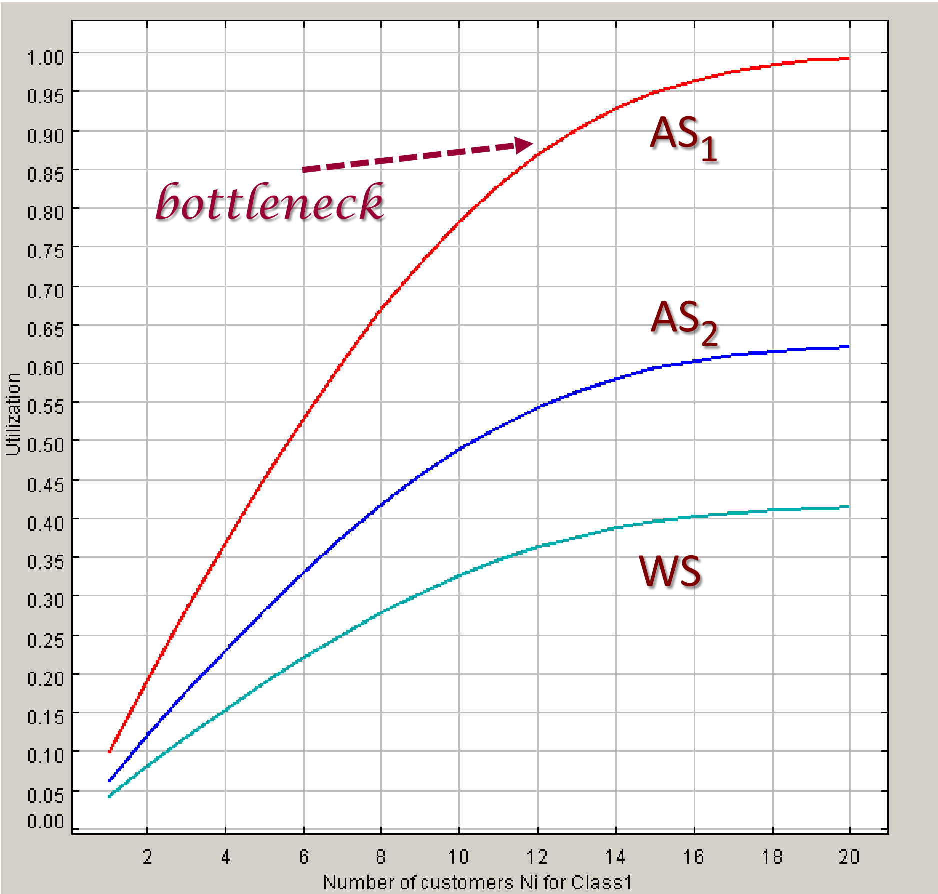
N = 16	N = 17	N = 18	N = 19	N = 20
1.992	2.098	2.191	2.293	2.402
2.022	2.130	2.228	2.332	2.449
1.961	2.066	2.153	2.253	2.355

System Response time R_o

distribution of response times

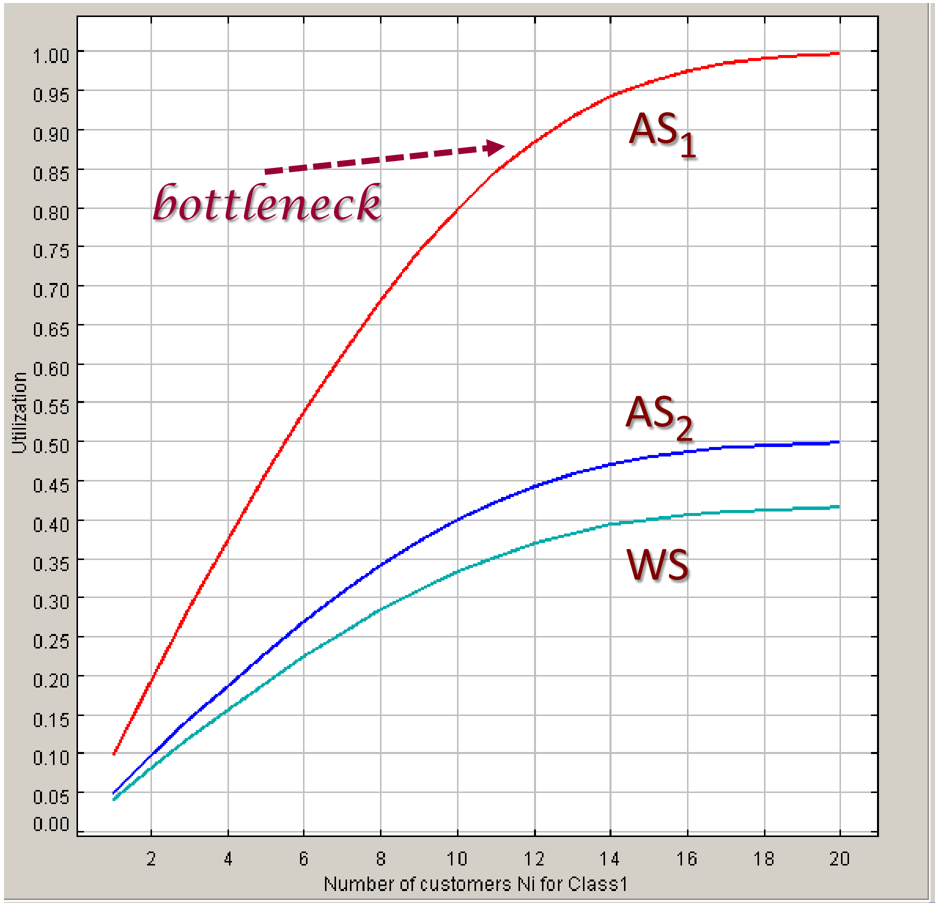


server Utilizations



with the original configuration

S_{AS2}=0.025sec



with server AS2 20% faster

S_{AS2}=0.020sec

improvements

- with the original configuration $X_0(20) = 8.32$ j/s $R_0(20) = 2.4$ sec
- with upgraded AS_2 of 20% $X_0(20) = 8.27$ j/s $R_0(20) = 2.42$ sec
  (the bottleneck is AS1) NO IMPROVEMENTS !!!
- with upgraded AS_1 of 20% $X_0(20) = 9.99$ j/s $R_0(20) = 1.99$ sec
  +20% INCREASE -17% DECREASE

⬇

improving any resource but the bottleneck __do not__ generate any performance gain with heavy workload

with N=40 users - asymptotic values

- original config. $N_0=20$ users $X_0(20)= 8.24$ j/s $R_0(20)= 2.43$ sec
- original config. $N_0=40$ users $X_0(40)= 8.24$ j/s $R_0(40)= 4.84$ sec

$U_{AS1}=1$ $U_{AS2}=0.62$

+0% ! *+99.1%*

the bottleneck is still on server AS1 that is saturated !

asymptotes

- $X_{max}=1/D_{max}=1/0.12=8.33$ j/s $R_{min}=N_0\ D_{max}=40 \times 0.12=4.8$ sec
- (Little law is $R=(N/X)-Z$, this R_{min} includes Z)

Equivalent model
with
Service Demands

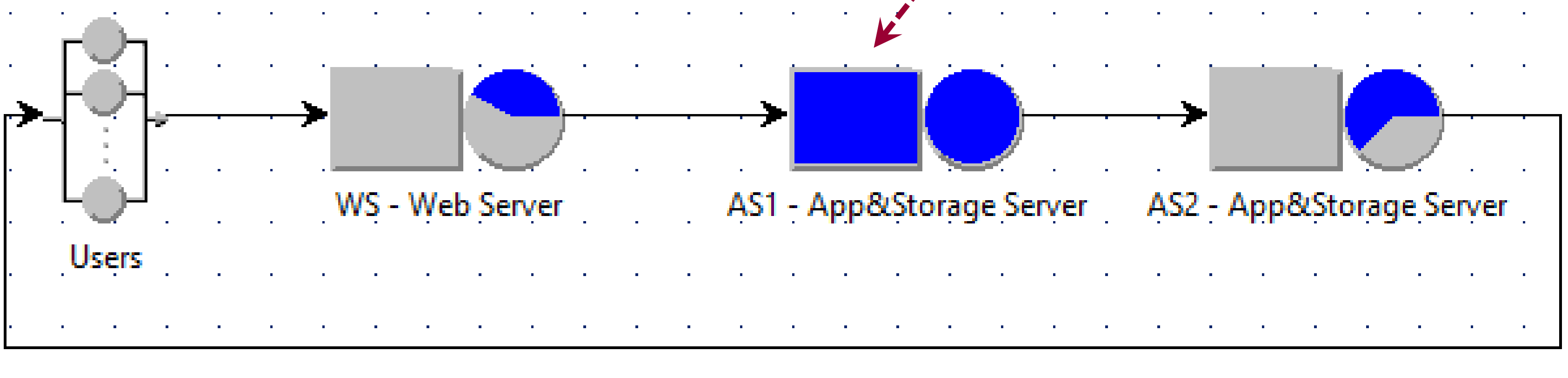
Sec.2.3

Sec.2.3 - model with Service Demands

Service demands: D_r

- $D_{WS} = V_{WS}\, S_{WS} = 0.05$sec
- $D_{AS1} = V_{AS1}\, S_{AS1} = 0.12$sec
- $D_{AS2} = V_{AS2}\, S_{AS2} = 0.075$sec

the bottleneck is still on server AS1 that is saturated !



Sec.2.3 - equivalent model performance metrics

utilization of resources

residence times

throughput of resources

SAME VALUES

Parameters used	R_0	X_0	U_{WS}	U_{AS1}	U_{AS2}	Rd_{WS}	Rd_{AS1}	Rd_{AS2}	X_{WS}	X_{AS1}	X_{AS2}
Visits & Service time (Fig. 2.7)	**2.4**	**8.32**	.419	.993	.618	.082	1.14	.184	82.13	49.32	24.74
Service demands (Fig. 2.15)	**2.4**	**8.31**	.415	.992	.622	.083	1.14	.185	8.31	8.31	8.31

requests

jobs

Optimal operating point of a server

Chapter 2 --- Sect. 2.4

open model
single class
tool used: JSIMg

Sect.2.4 - the problem (M/M/1) [Giessler, Kleinrock, 78, 79]

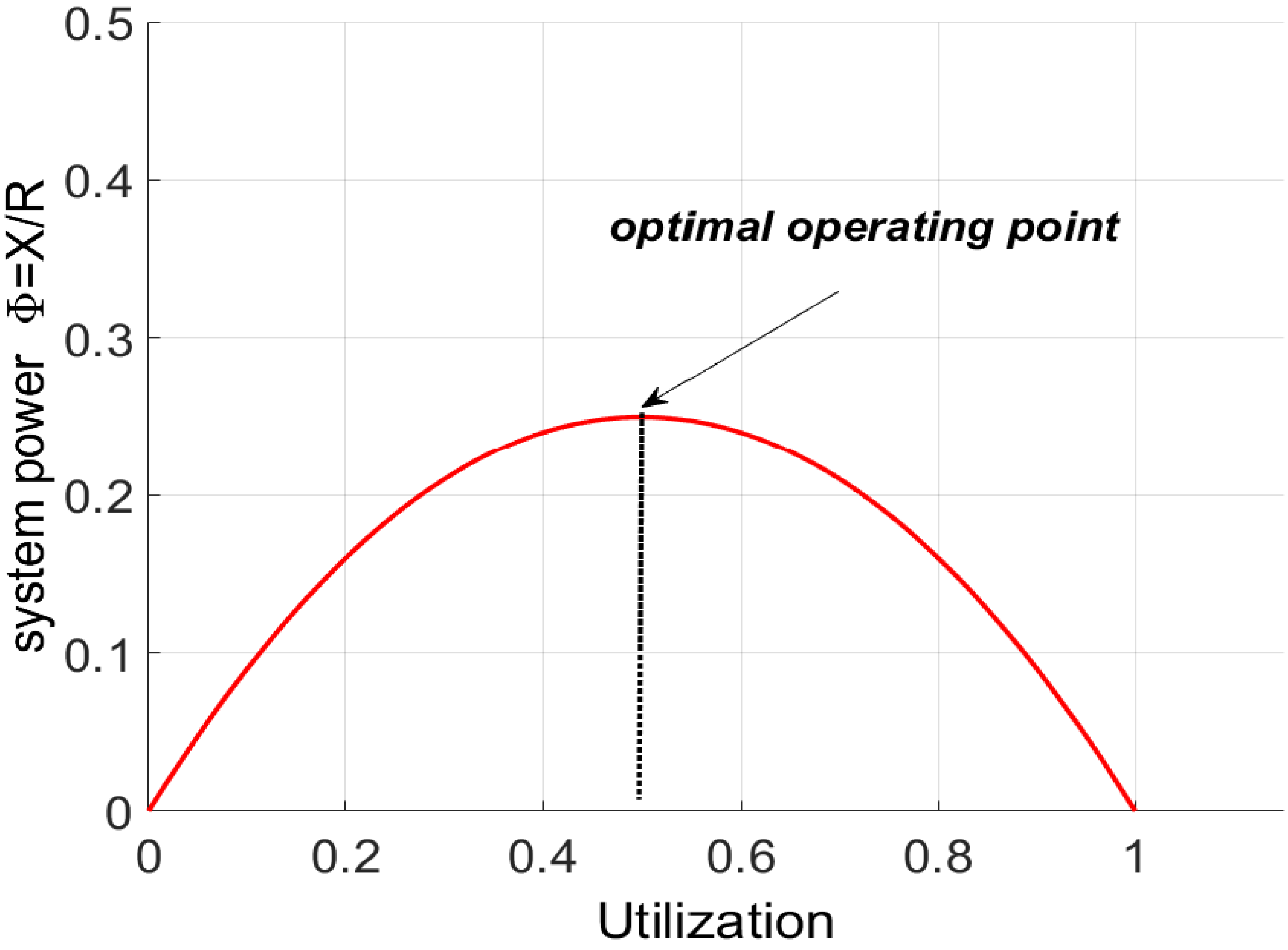
- system power: $\Phi = \frac{X}{R}$ in M/M/1 $\Phi = \frac{\lambda}{R} = \frac{\lambda\,(1 - \lambda S)}{S}$

- identify the λ^{opt} that maximizes the system power, i.e., the throughput is maximum with the minimum response time

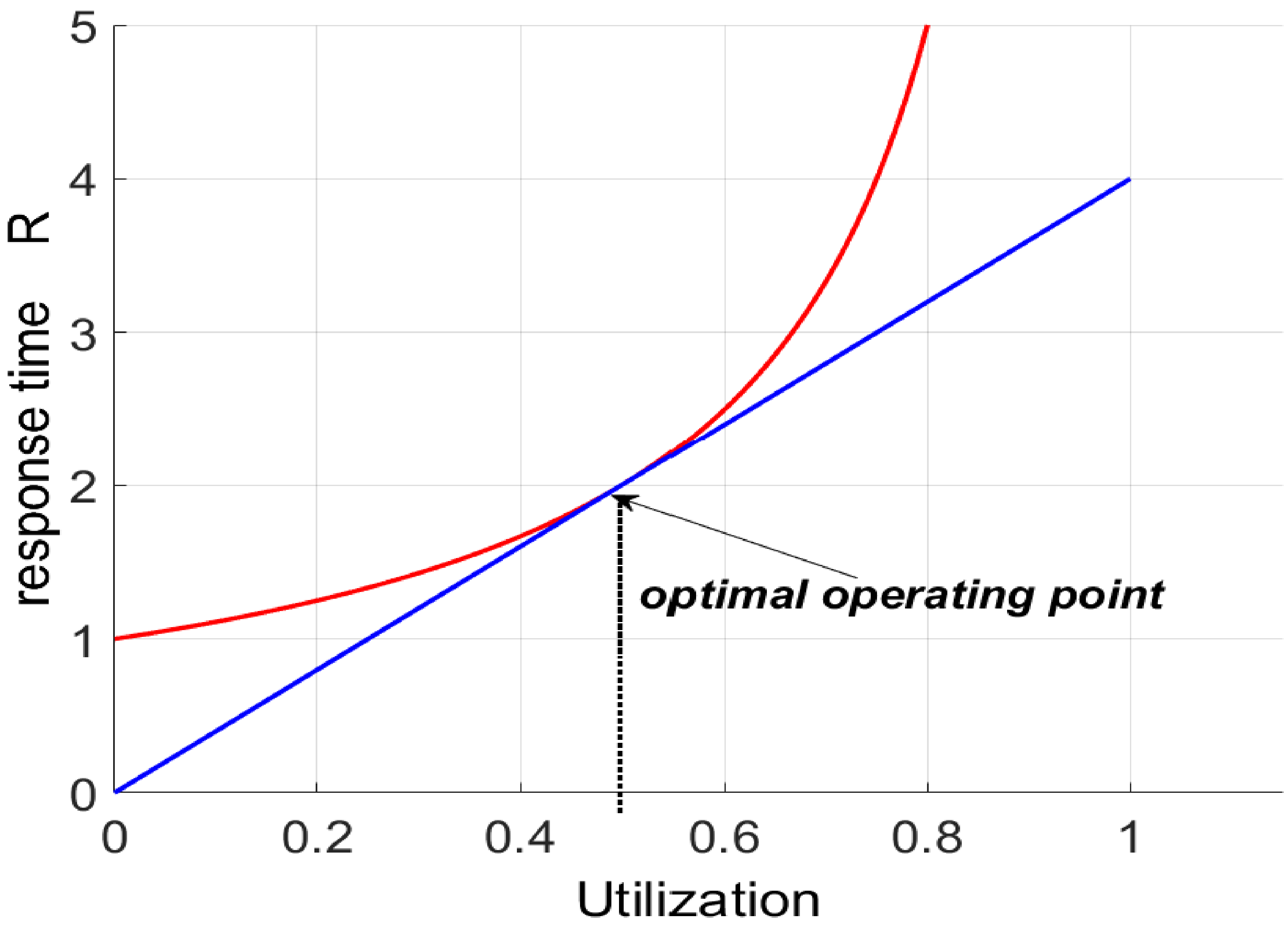
$$\lambda^{opt} = \frac{1}{2}\,\frac{1}{S}$$

$$R^{opt} = \frac{S}{1 - \lambda^{opt} S} = 2S \qquad U^{opt} = \lambda^{opt} S = 0.5 \qquad N^{opt} = \frac{0.5}{1 - 0.5} = 1\ req$$

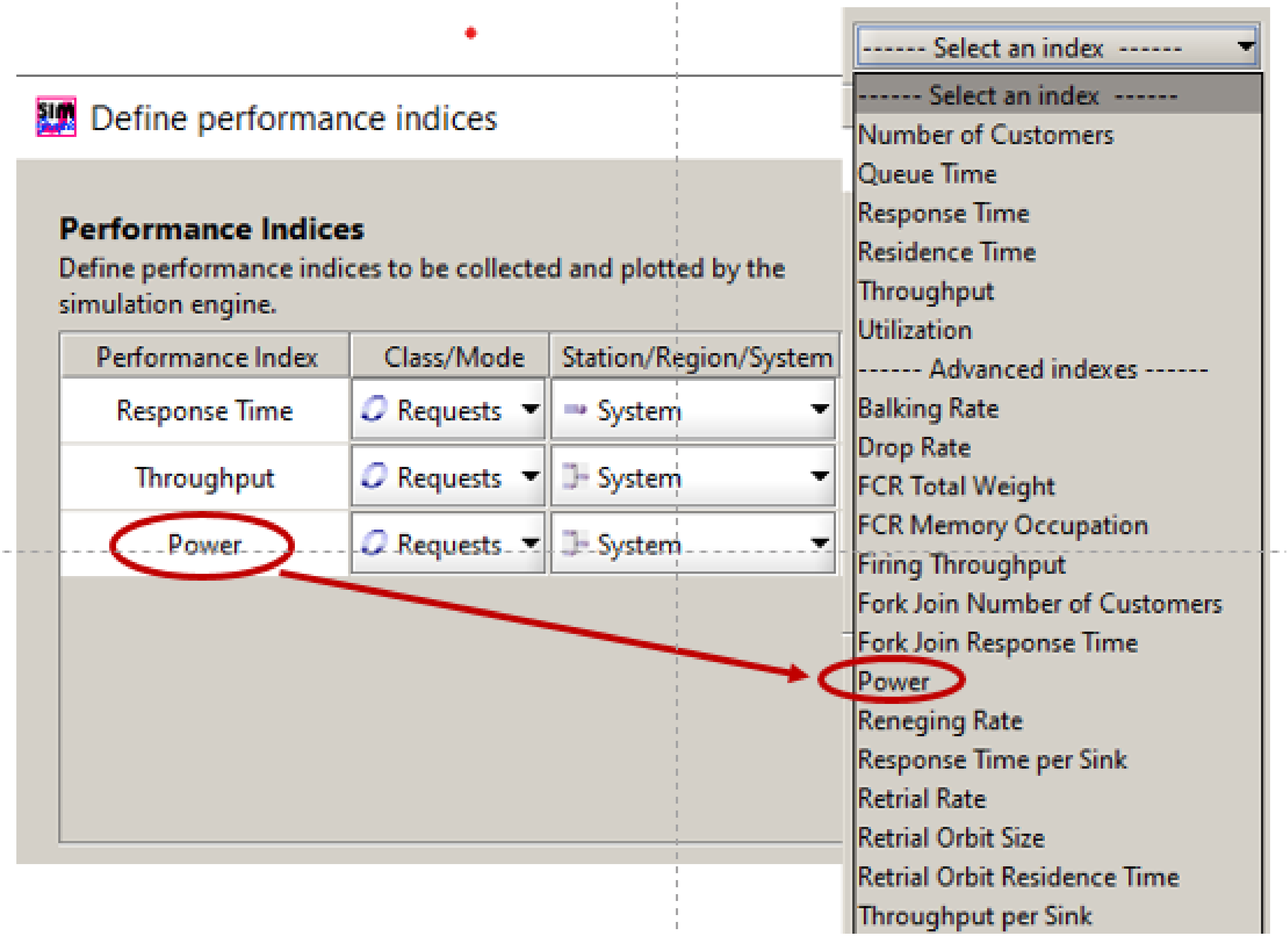
system power Φ, S=1, (analytical result)



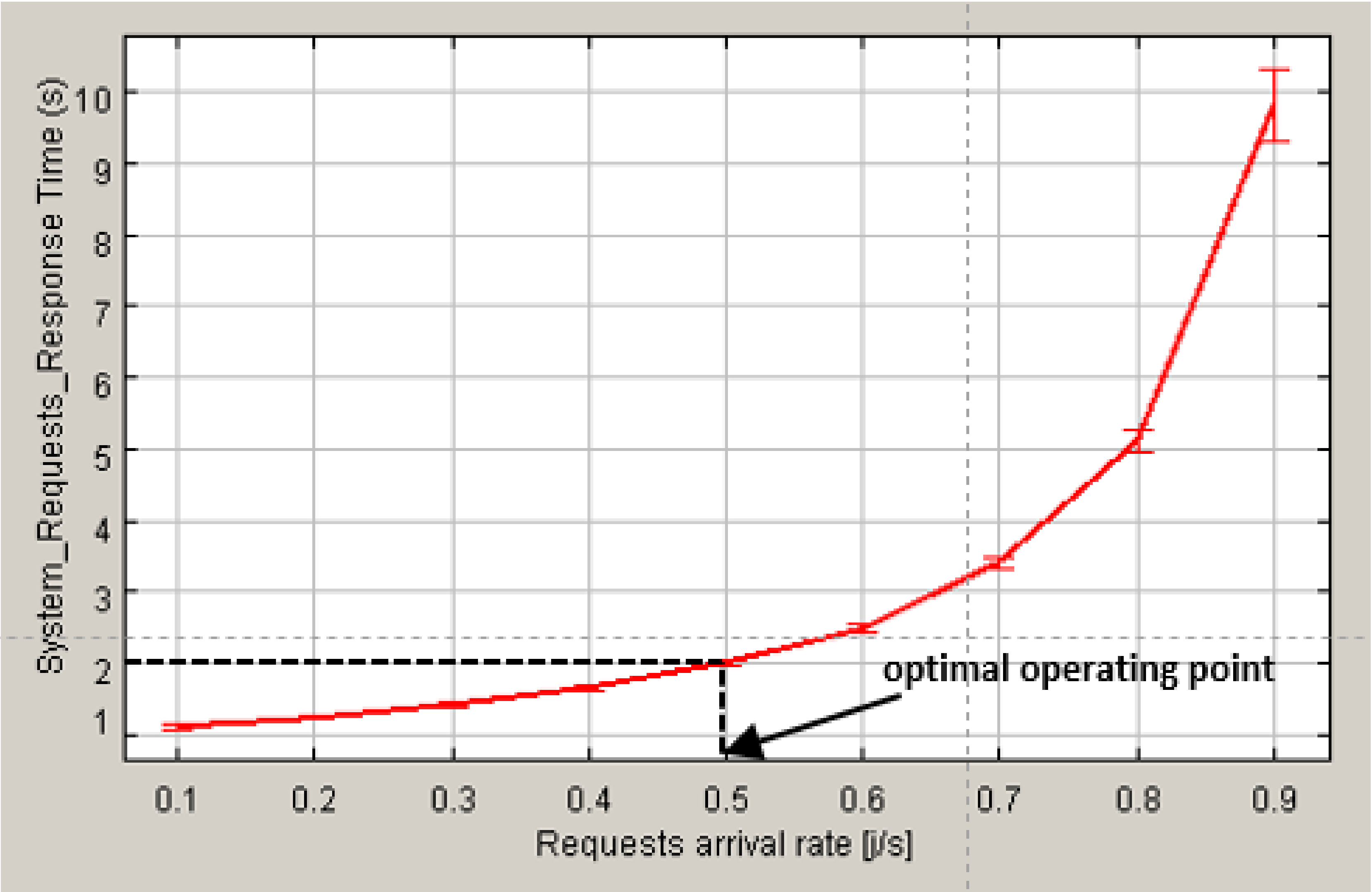
optimal load (analytical result)



selection of the performance indices (JSIMg)



optimal operating point (simulation, JSIMg)



Impact of
Bottleneck migration

Chapter 3 --- Sect. 3.2

closed model
heterogeneous (2 class) workload
tools used: JMVA

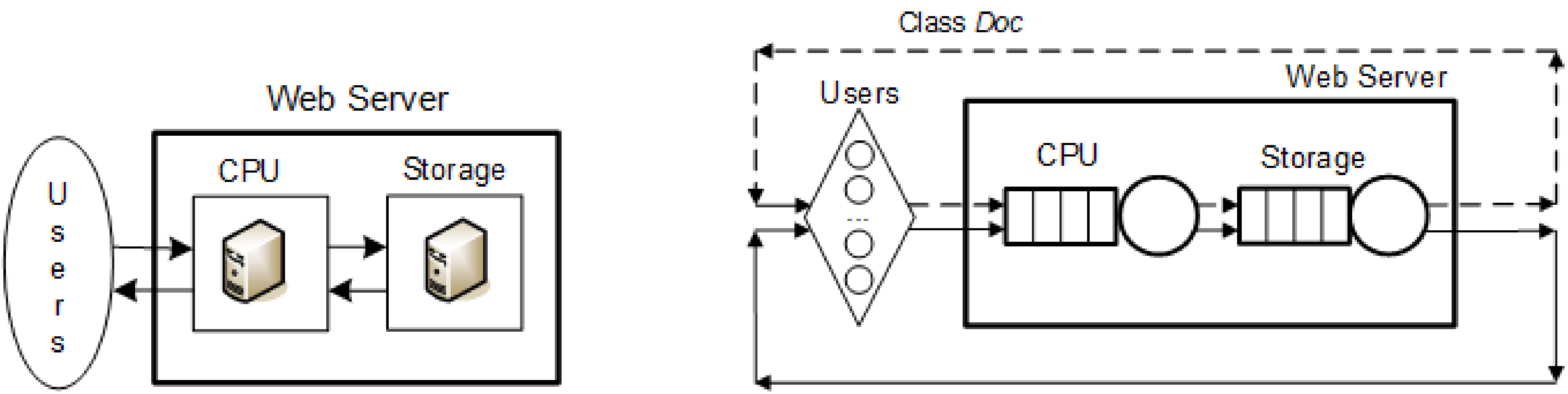
Sect.3.2 - the problem

- web server accessed by administrative staff and graduate students
- two classes of customers with different resource requests and performance objectives
  - class-Adm: management of the administrative procedures concerning the students curricula (tuition fees payments, courses attended, grades obtained, ...
  - class-Doc: management of the course materials (slides, notes, homeworks, exams, ...)
- capacity planning: performance forecast with the increase in class-Doc customers

workload

- Service **D**emands, resource i , class-r $D_{i,r} = V_{i,r} \times S_{i,r}$
- system population: $\underline{N}=\{N_{0,Adm}, N_{0,Doc}\}$ from {20,5} to {20,280}
- unbalanced population growth: Doc from 5 to 280 ($N_0=25 \div 300$)
- population mix: $\underline{\beta}=\{\beta_{Adm},\beta_{Doc}\}$, $\{N_{0,Adm}/N,\ N_{0,Doc}/N\}$ fraction of cust. per class

Sect.3.2 - the system considered



(a) (b)

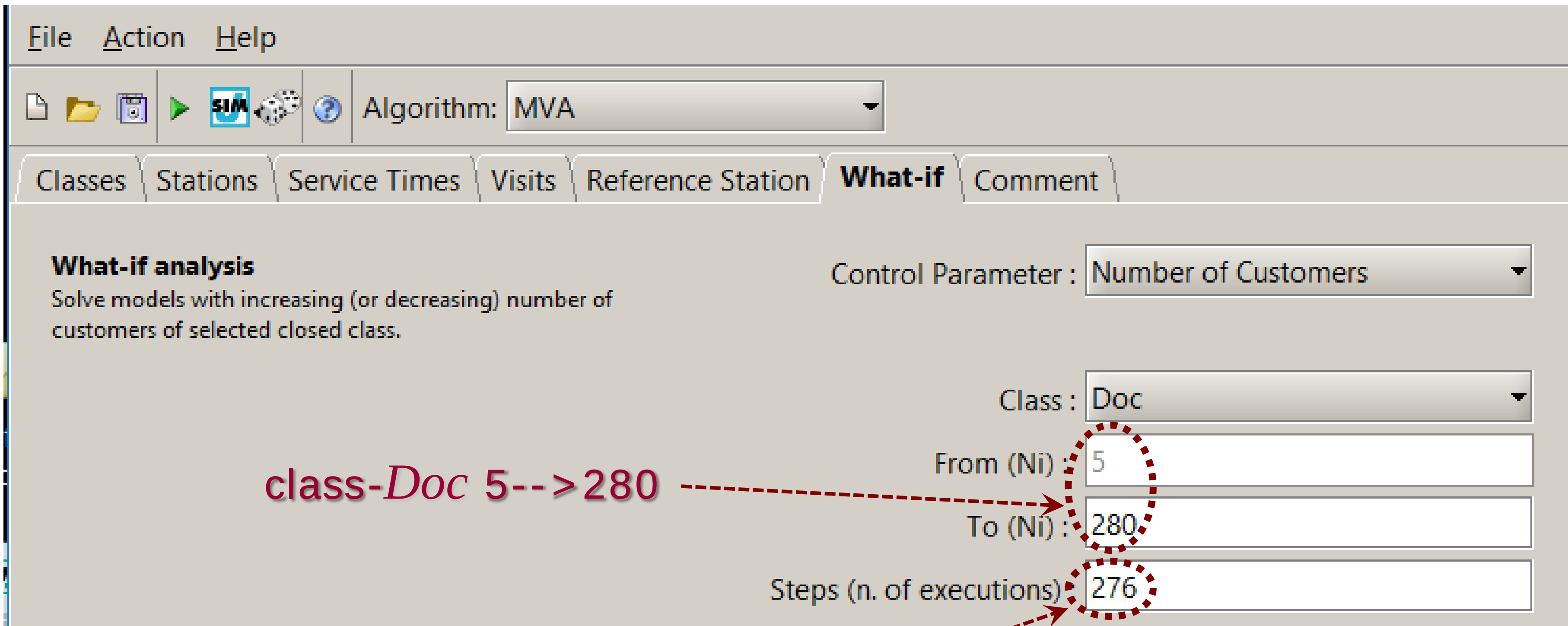
Service Demands [sec]

Resources (Stations)	Two classes	
	Adm	*Doc*
Users Think time	3	10
CPU	0.20	0.100
Storage	0.050	0.60

class *Adm* bottleneck

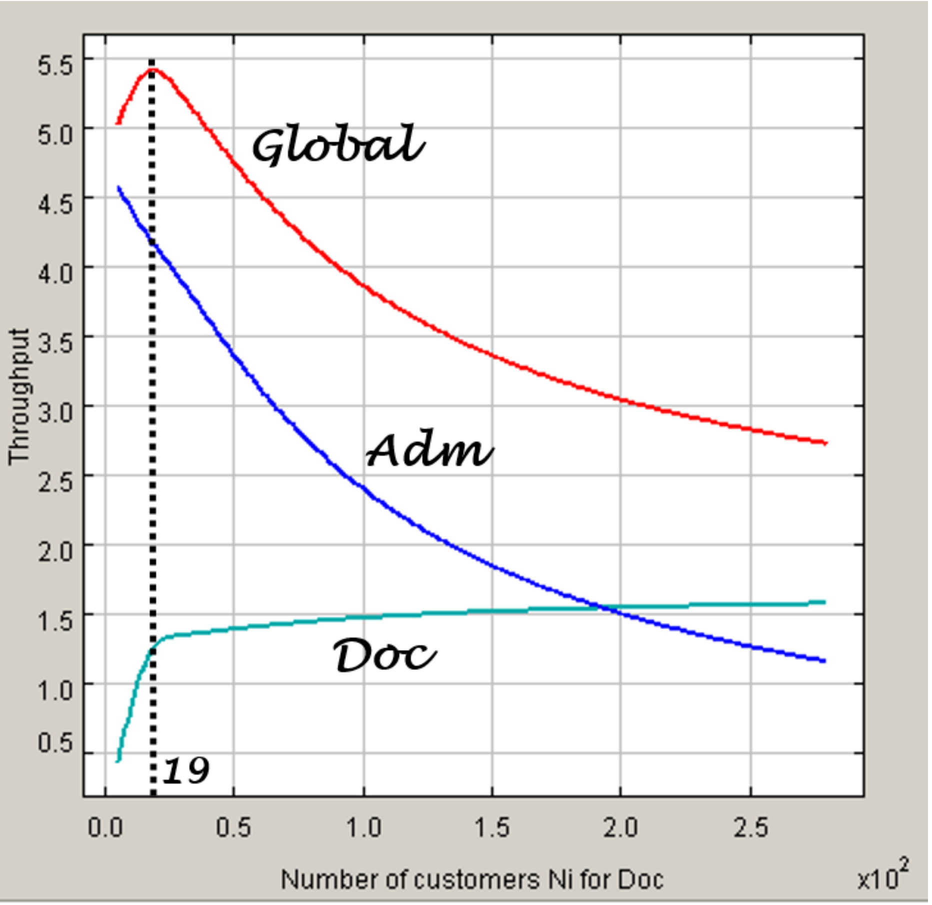
class *Doc* bottleneck

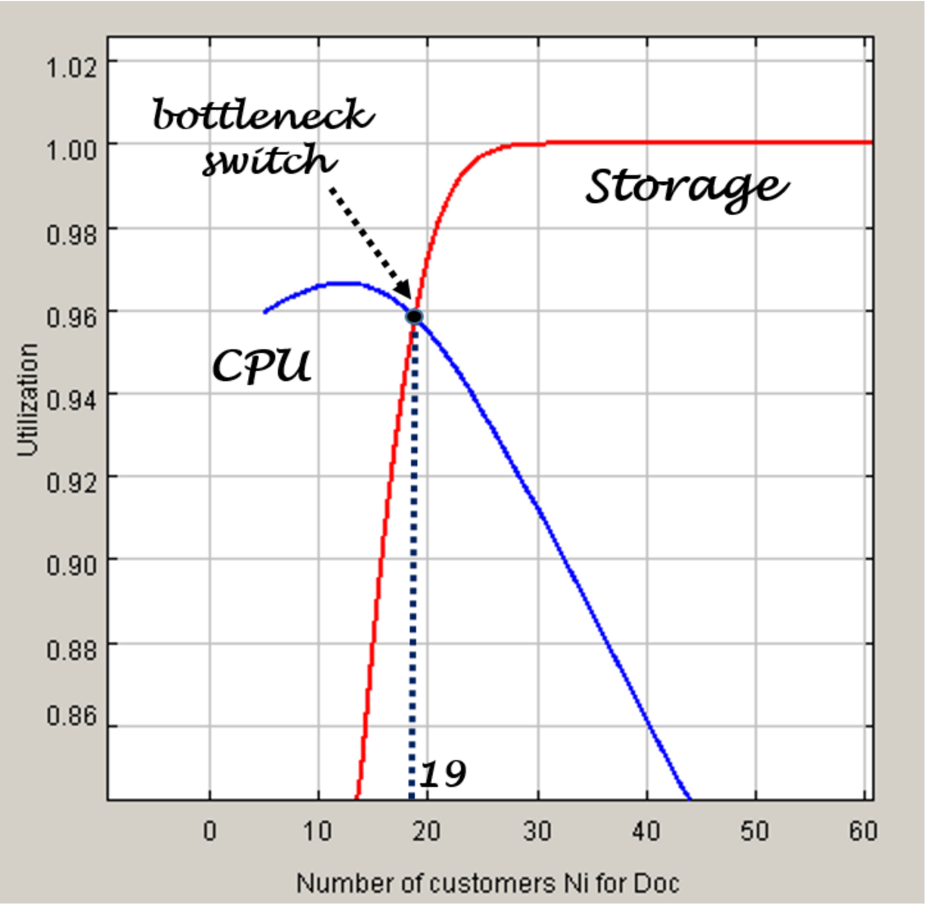
What-if analysis: unbalanced population growth



total number of models

increase of class-*Doc* only (5 ÷ 280)

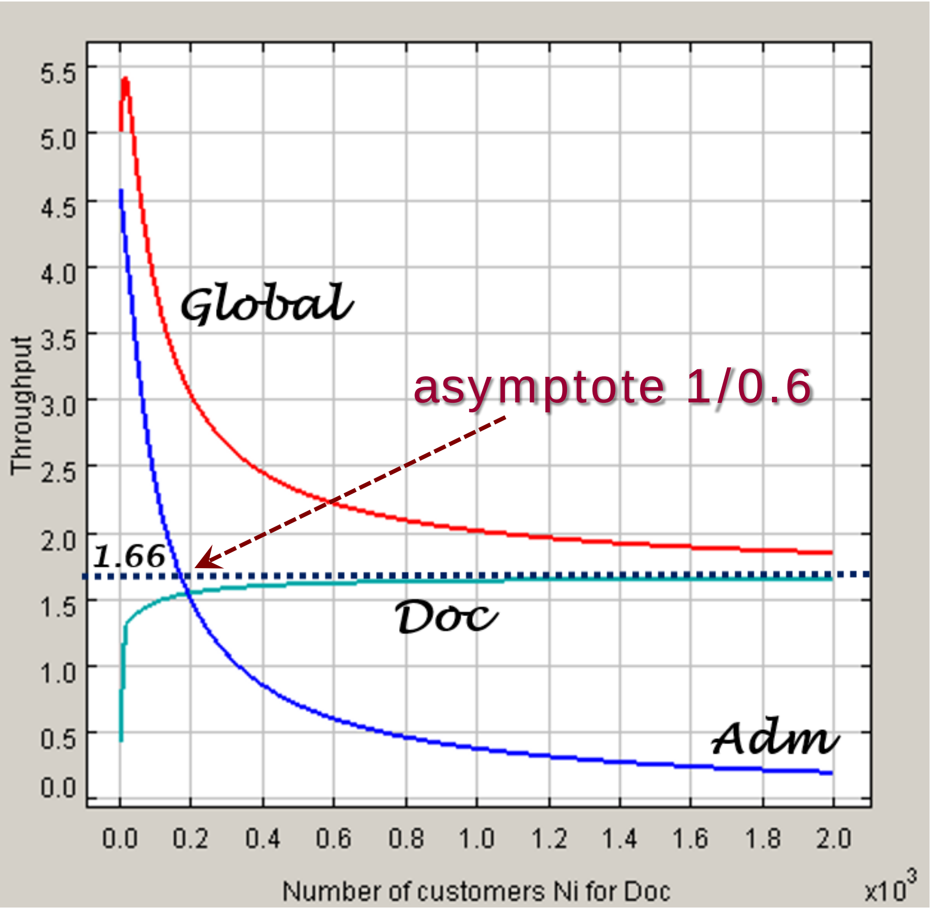




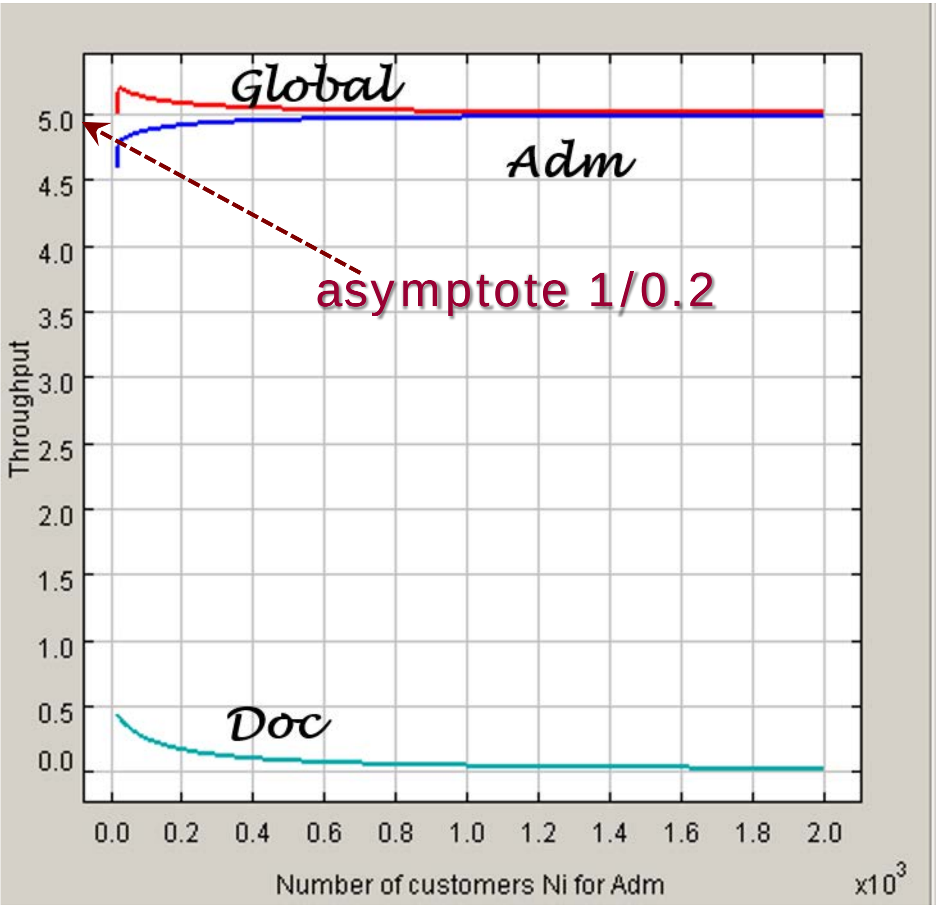
system throughput

utilizations

system throughput with increase of one class only



class-*Doc* only (5 ÷ 2000)



class-*Adm* only (20 ÷ 2000)

Performance Optimization

of a

Data Center

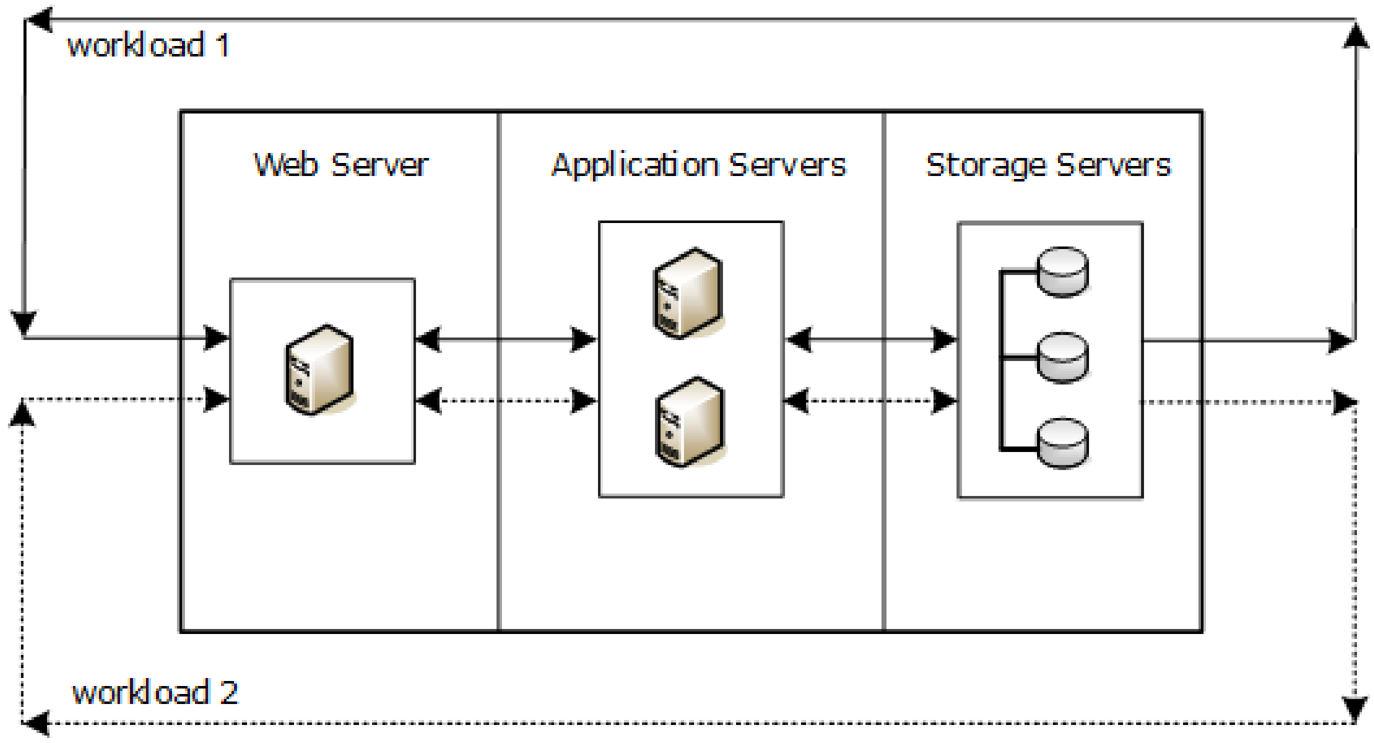
Chapter 3 --- Sect. 3.3

closed model
heterogeneous (multiclass) workload
tools used: JMVA, JABA

Sect.3.3 - the system considered

- six servers of the data center are utilized by business critical apps that access to sensitive data stored
- for security reasons the accesses are allowed only to a restricted (constant) number of employees
- two classes of requests with different resource loads and performance requirements
- class-1 : GUI management, business logic computation
- class-2 : data processing (store, search, update, download, ...)
- to reduce the number of parameters (and to simplify obtaining their values) we have parameterized the model in terms of global loads to resources, i.e., with the service demands $D_{r,c}$
- $D_{r,c}$ represent the global amount of service time required by a complete execution of a class-c request to resource r

the data center structure



objectives

- the number of users is expected to increase from 100 to 200. Does the current configuration support this increase without saturating resources? (bottleneck identification)
- it is required that the per-class target of response times ($R_{0,1} <= 8$sec , $R_{0,2} <= 12$sec) set for 100 users must also be satisfied for 200 users (with 100 users of class-1 and 100 users of class-2)
- the fraction of requests in execution of the two classes (i.e., the population mix) vary over time
- identify the actions that improve performance and the population mixes that maximize the System Throughput X_0 and minimizes the System Response time R_0

workload parameters

- Service **D**emands, resource i , class-r $D_{i,r} = V_{i,r} \times S_{i,r}$
- global number of customers : N=200
- system population: $\underline{N}=\{N_{0,1}, N_{0,2}\}$, from {0,200} to {200,0}
- population mix: $\underline{\beta}=\{\beta_1,\beta_2\}$, $\{N_{0,1}/N, N_{0,2}/N\}$ fraction of cust. per class
- $\underline{\beta}$ variable: performance forecast with all the mixes

Service demands (original system), ms

Classes | Stations | **Service Demands** | Reference Station | What-if | Comment

Service Demands

Input service demands of each station and class.
If the station is "Load Dependent" you can set the service demands for each number of customers by double-click on "LD Settings..." button.
Press "Service Times and Visits" button to enter service times and visits instead of service demands.

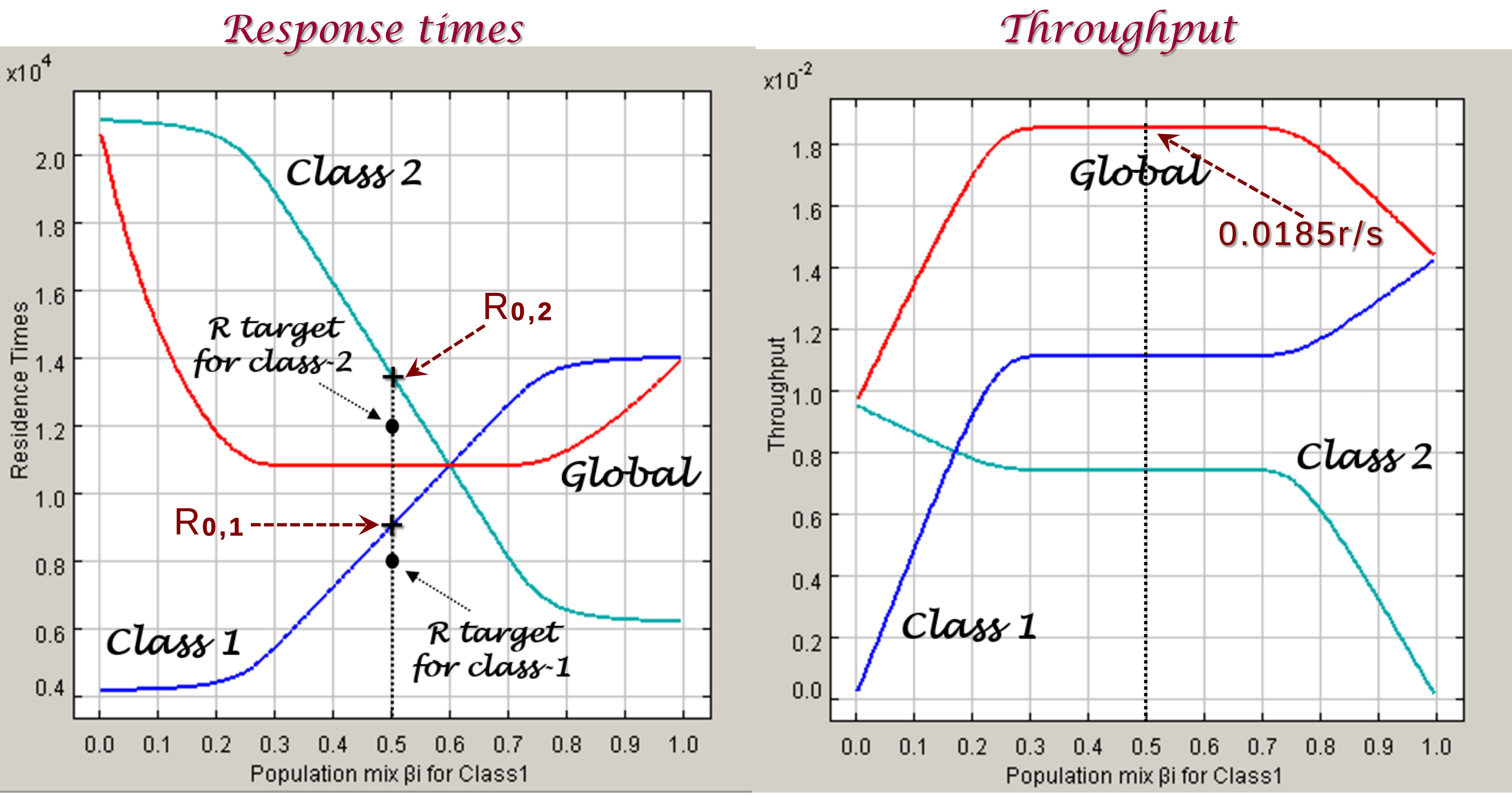
*	Class1	Class2
Web_Server	12.0000	7.0000
App_Server1	14.0000	20.0000
App_Server2	23.0000	14.0000
Storage1	20.0000	105.0000
Storage2	70.0000	30.0000
Storage3	25.0000	33.0000

natural bottleneck of class 1 (Storage 2)

unbalanced demands

natural bottleneck of class 2 (Storage 1)

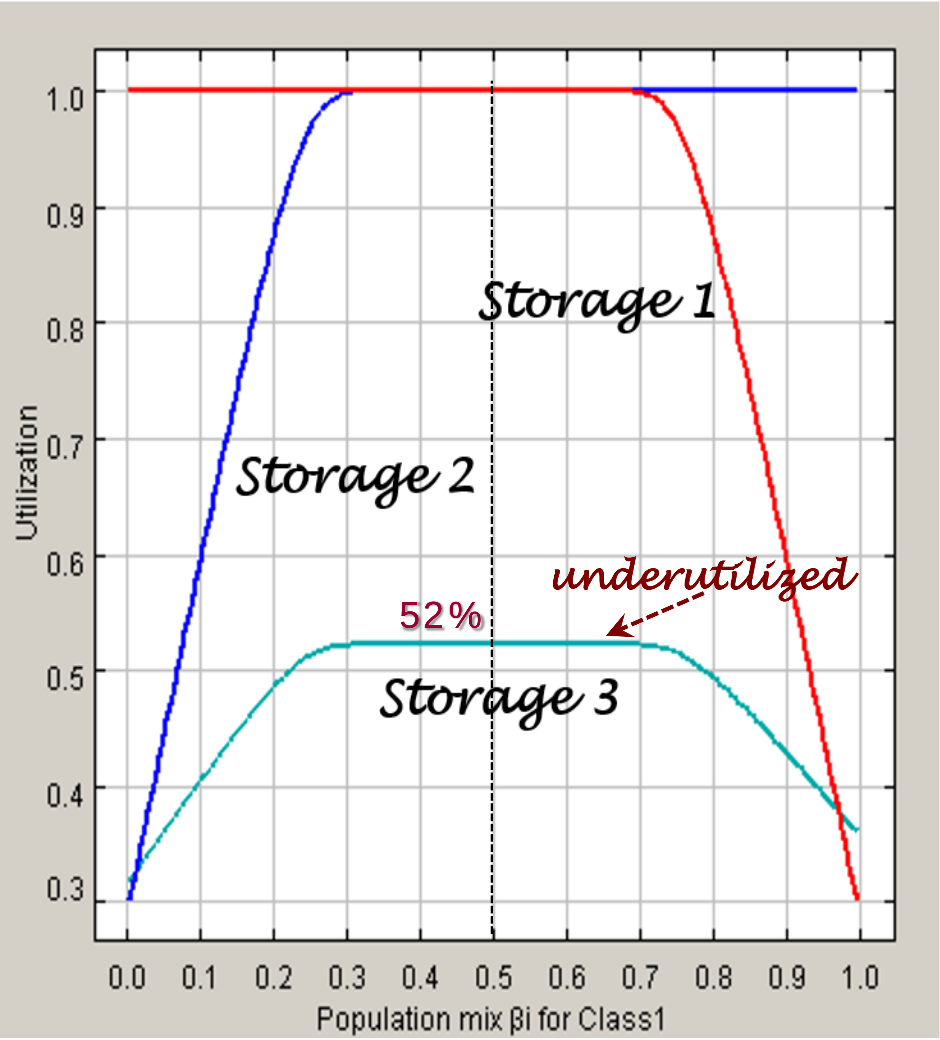
R_0 and X_0 (per-class and global) for all program mix, N=200



R_0 target times with β={0.5,0.5}: 8s for class1, 12s for class2

NOT SATISFIED --> $R_{0,1}$= 9 sec > 8 sec $R_{0,2}$= 13.5 sec > 12sec

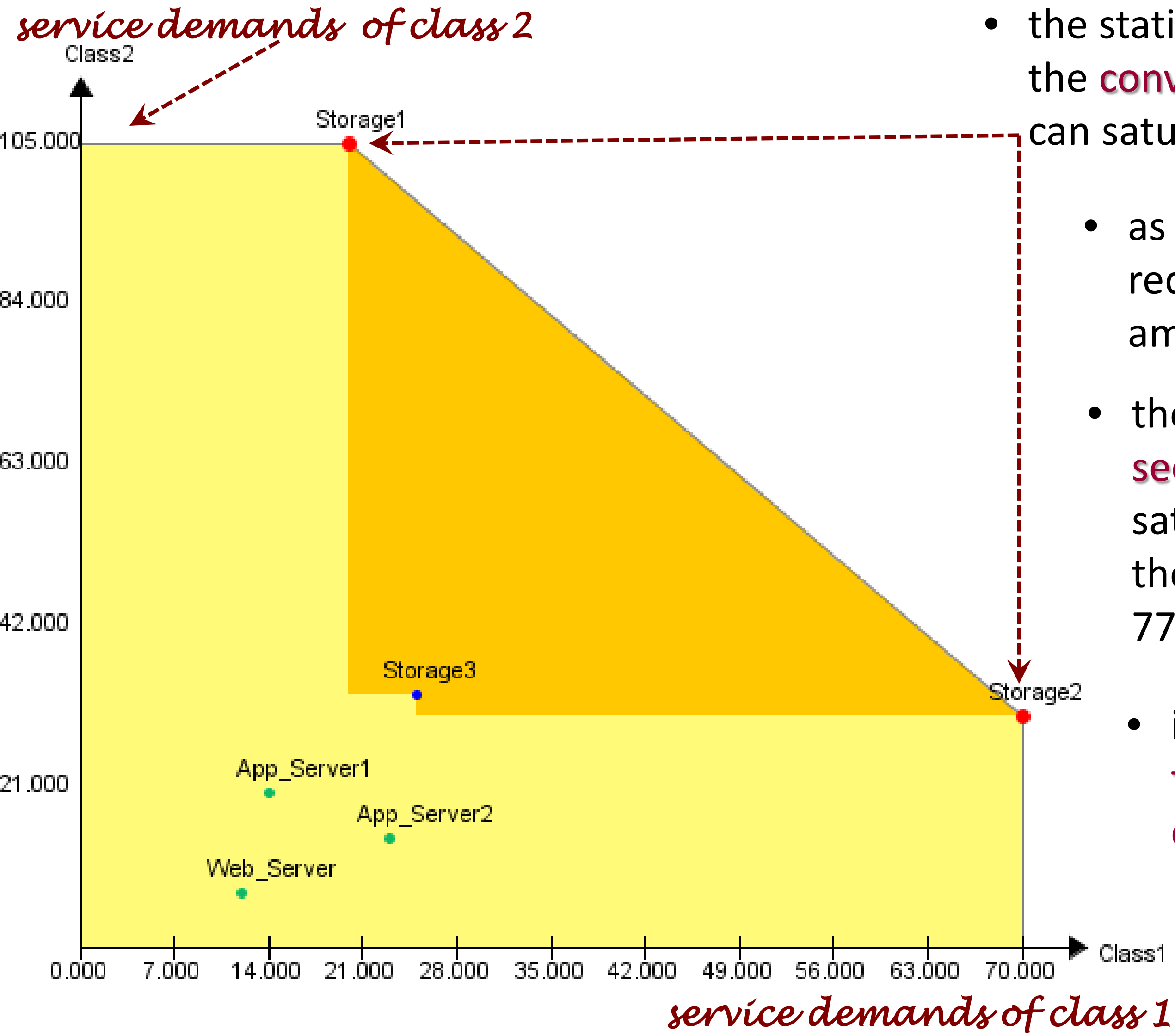
utilization of the three Storage servers (for all the mixes)



Original system

- the bottleneck is Storage1 when the number of class1 requests in execution is <30%, it is Storage2 when this number is >70%
- the utilizations of the three Storages are unbalanced, with $\underline{\beta}$={0.5,0.5} Storage1 and Storage2 saturate (100%) while Storage3 is 52% !
- the asymptotic utilizations are constant in the common saturation sector

potential bottlenecks - JABA



- the stations that lie on the boundary of the convex hull of the service demands can saturate (Storage1,Storage2)
  - as a function of the mix of requests the bottleneck migrate among them
  - there is as a common saturation sector, i.e., a set of mixes that saturate both the bottlenecks at the same time: from 22.2% to 77.8% of class1 [JABA]
    - in this sector the global response time and the throughputs are constant for all the mixes (see [3,15])

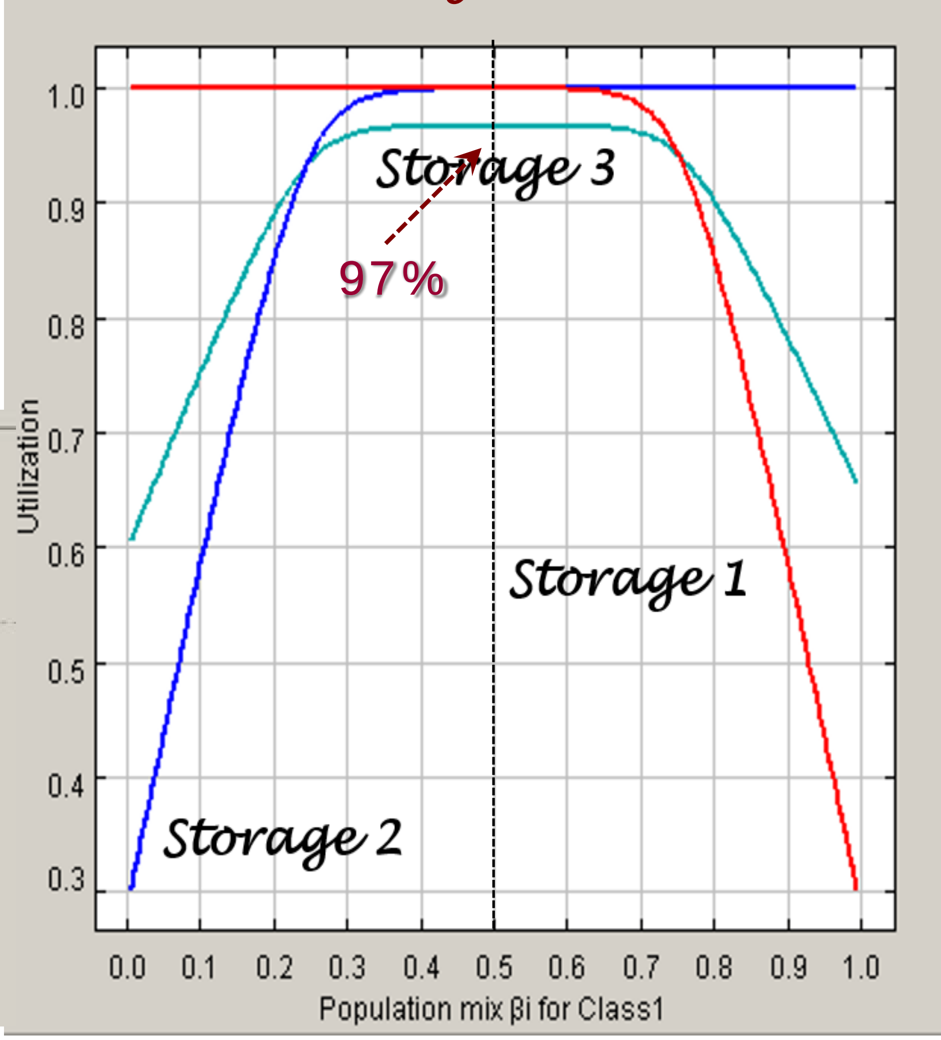
utilization of Storage servers in the balanced system

- data/files migration between Storages
- the global service demand of the three Storages is the same (283 ms) but their utilizations are almost balanced (Util. of Storage3 is 0.97)

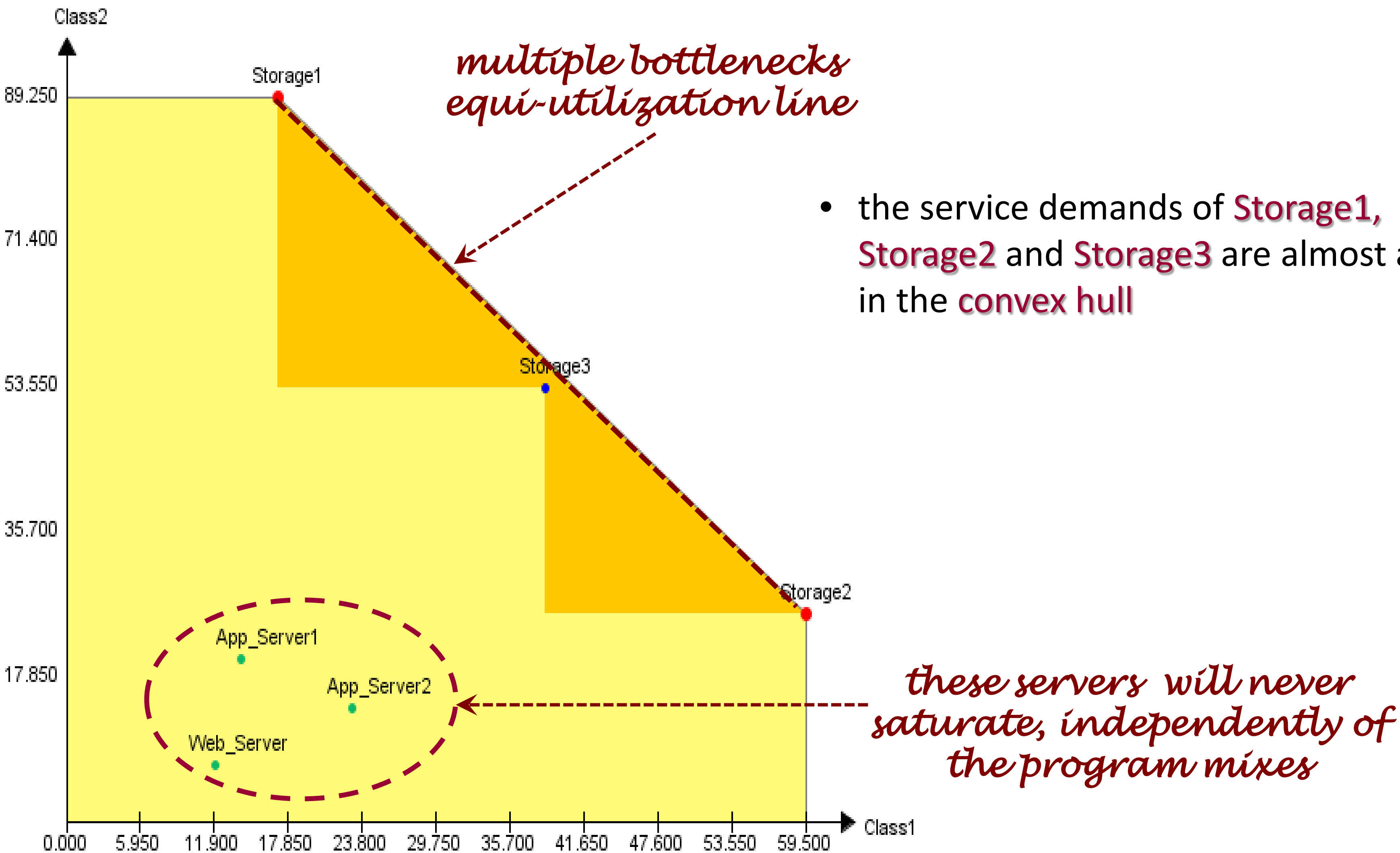
new Service Demands

*	Class1	Class2
Web_Server	12.0000	7.0000
App_Server1	14.0000	20.0000
App_Server2	23.0000	14.0000
Storage1	17.0000	89.2500
Storage2	59.5000	25.5000
Storage3	38.5000	53.2500

Utilizations



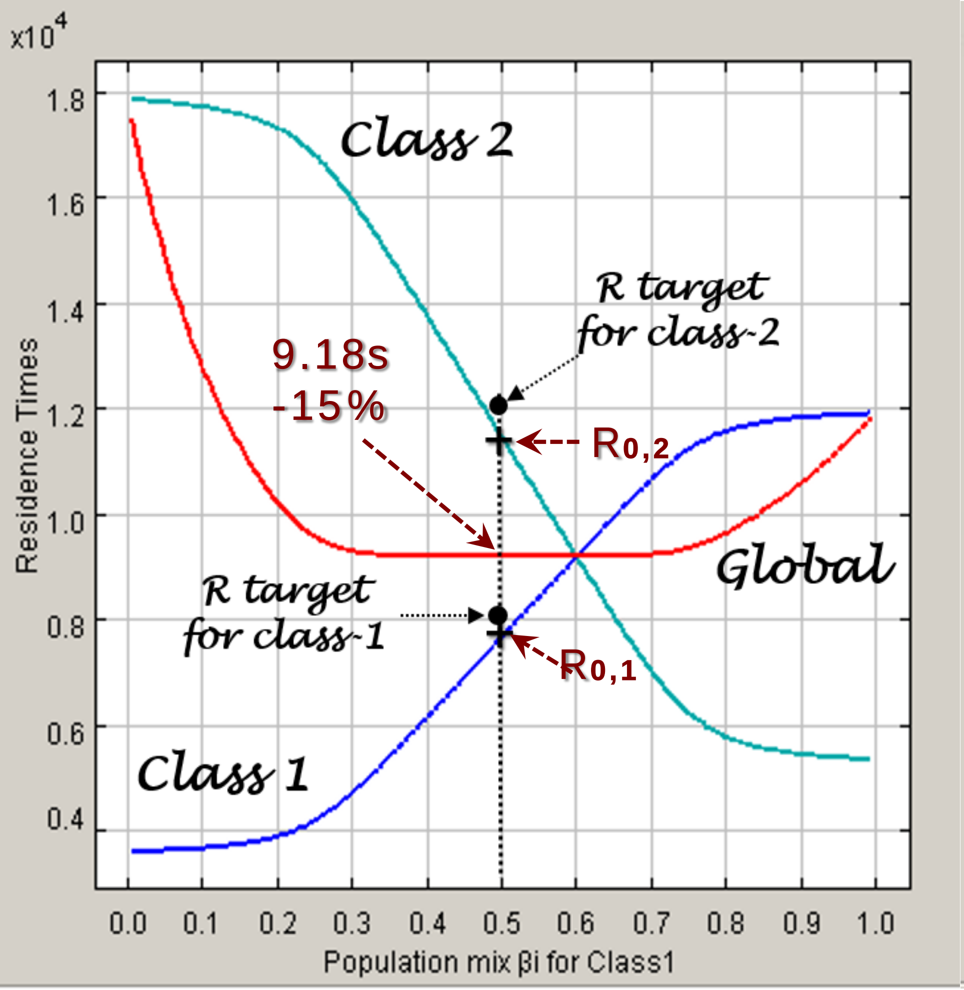
service demands of the balanced system - JABA



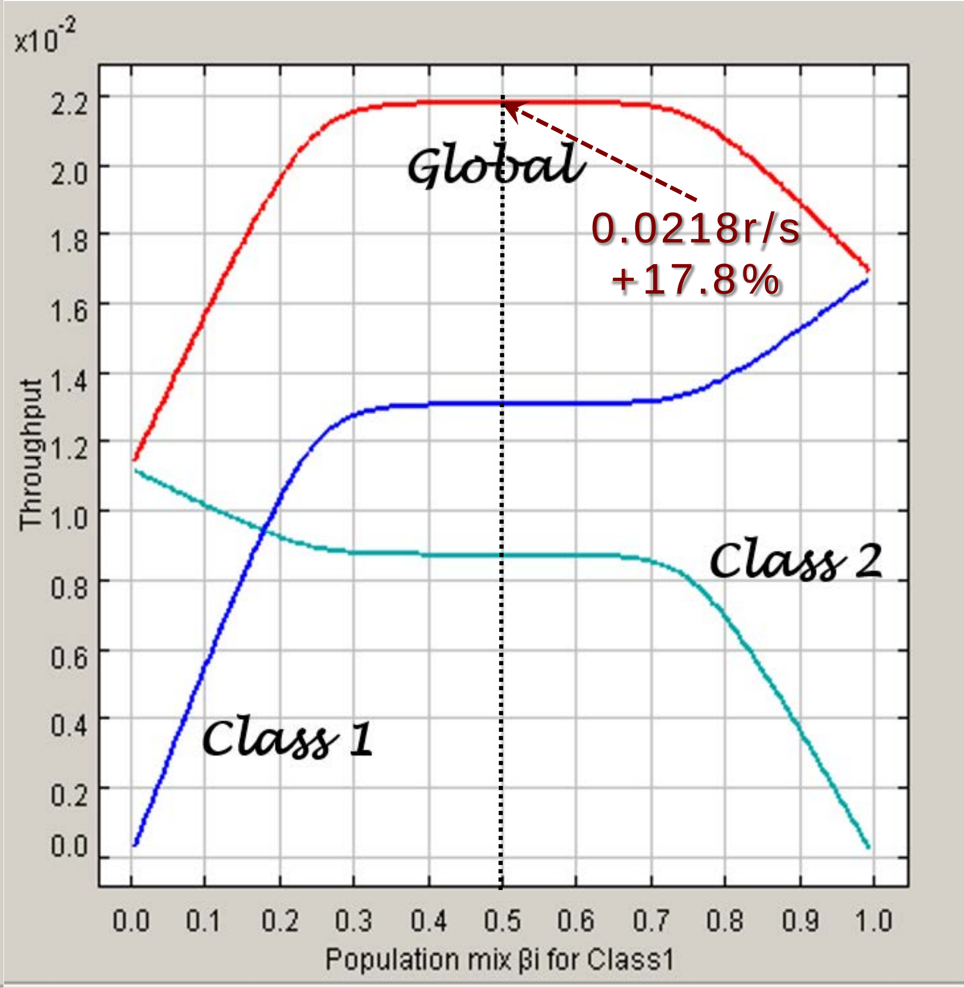
- the service demands of Storage1, Storage2 and Storage3 are almost aligned in the convex hull

R_0 and X_0 (per-class and global) of the balanced system, N=200

Response times



Throughput



with β={0.5,0.5} --> $R_{0,min}$ =10.8→9.18sec $X_{0,max}$ =0.0185→0.0218 req/ms

targets of R_0 are SATISFIED --> $R_{0,1}$= 7.65 <8sec $R_{0,2}$= 11.47 <12 sec

Variability of

Interarrival and Service times

Chapter 4 - Sect. 4.2 - Sect. 4.3

open models with different distributions
Deterministic, Hypo-exponential
Exponential, Hyper-exponential
tool used: JSIMg

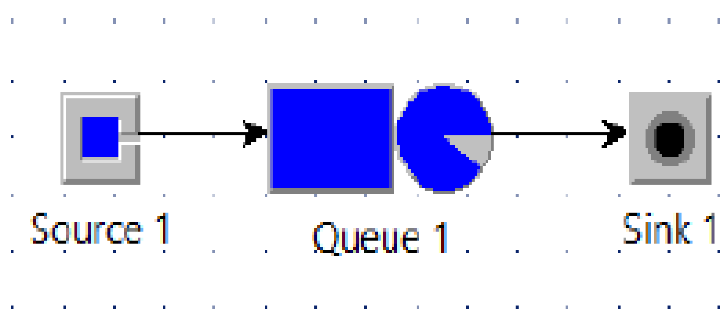
Sect.4.2 – the problem

- evaluate the impact of **variability of Interarrival times** on the performance
- the system consists of a queue station with service times **exponentially** distributed
- we consider **five models** with different distributions of Interarrival times with the same mean and increasing variance, with **coefficients of variation from 0 to 10**
- the request Arrival rate λ varies from **0.1 to 0.9 req/sec**

Variability of Interarrival times – JSIMg

Service times:

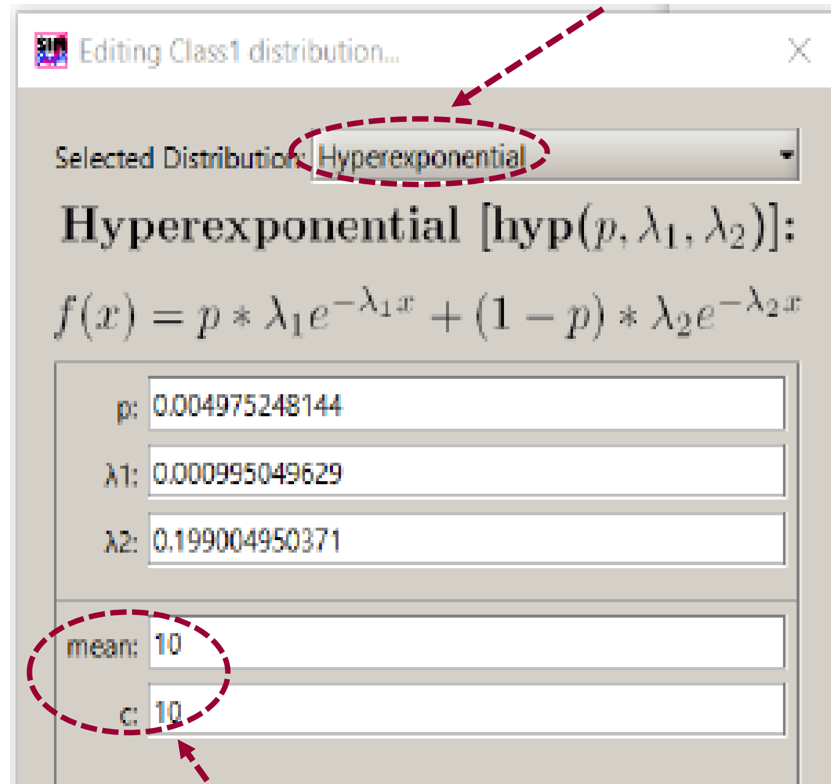
S=1sec Exponential (cv=1) for the five models



Interarrival times distributions:

- Deterministic cv=0
- Hypo-exponential cv=0.5
- Exponential cv=1
- Hyper-exponential cv=5
- Hyper-exponential cv=10

selection of the distribution

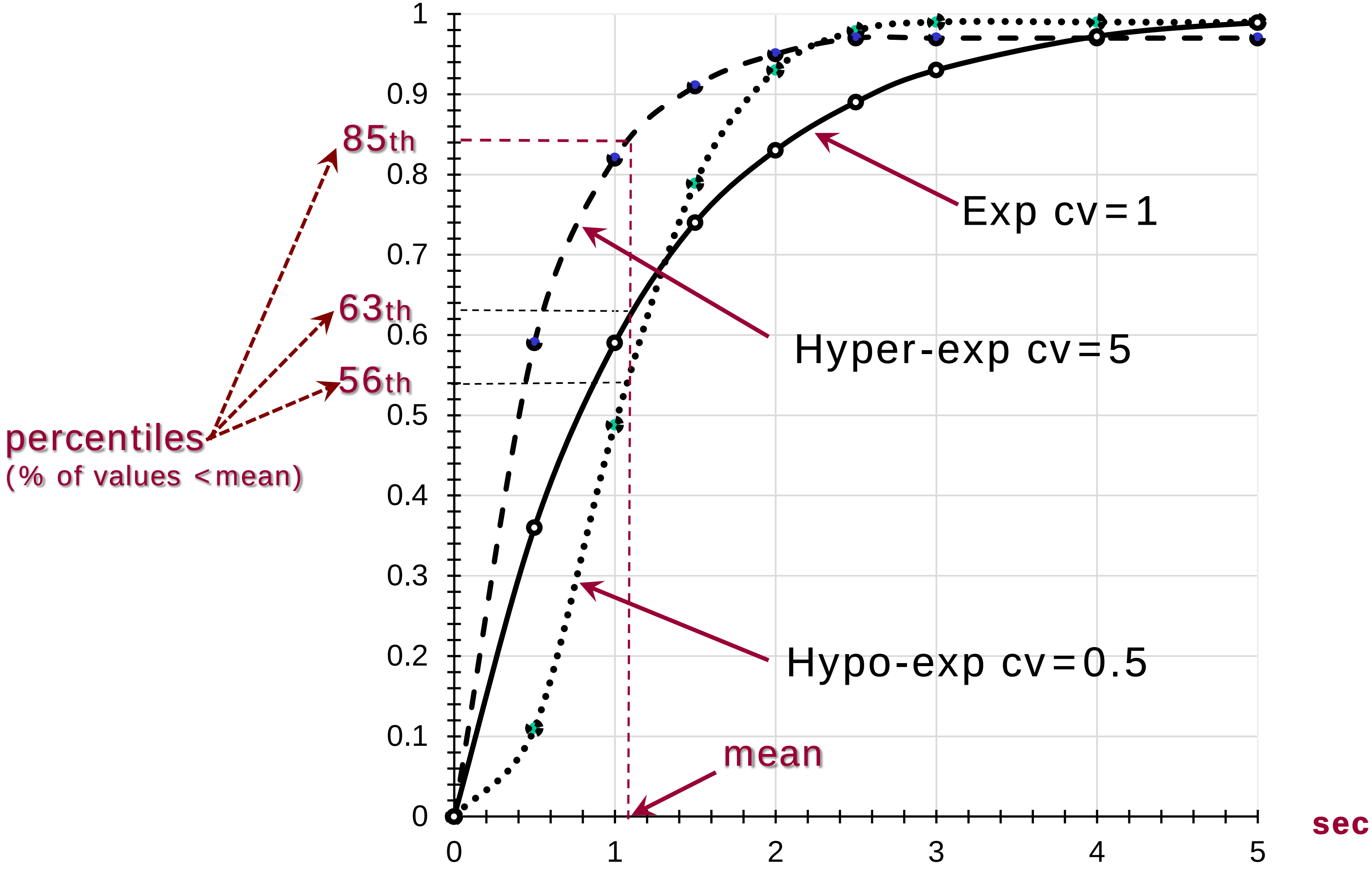


(b) Parameters of the Hyper-exponential distribution

mean and coefficient of variation cv

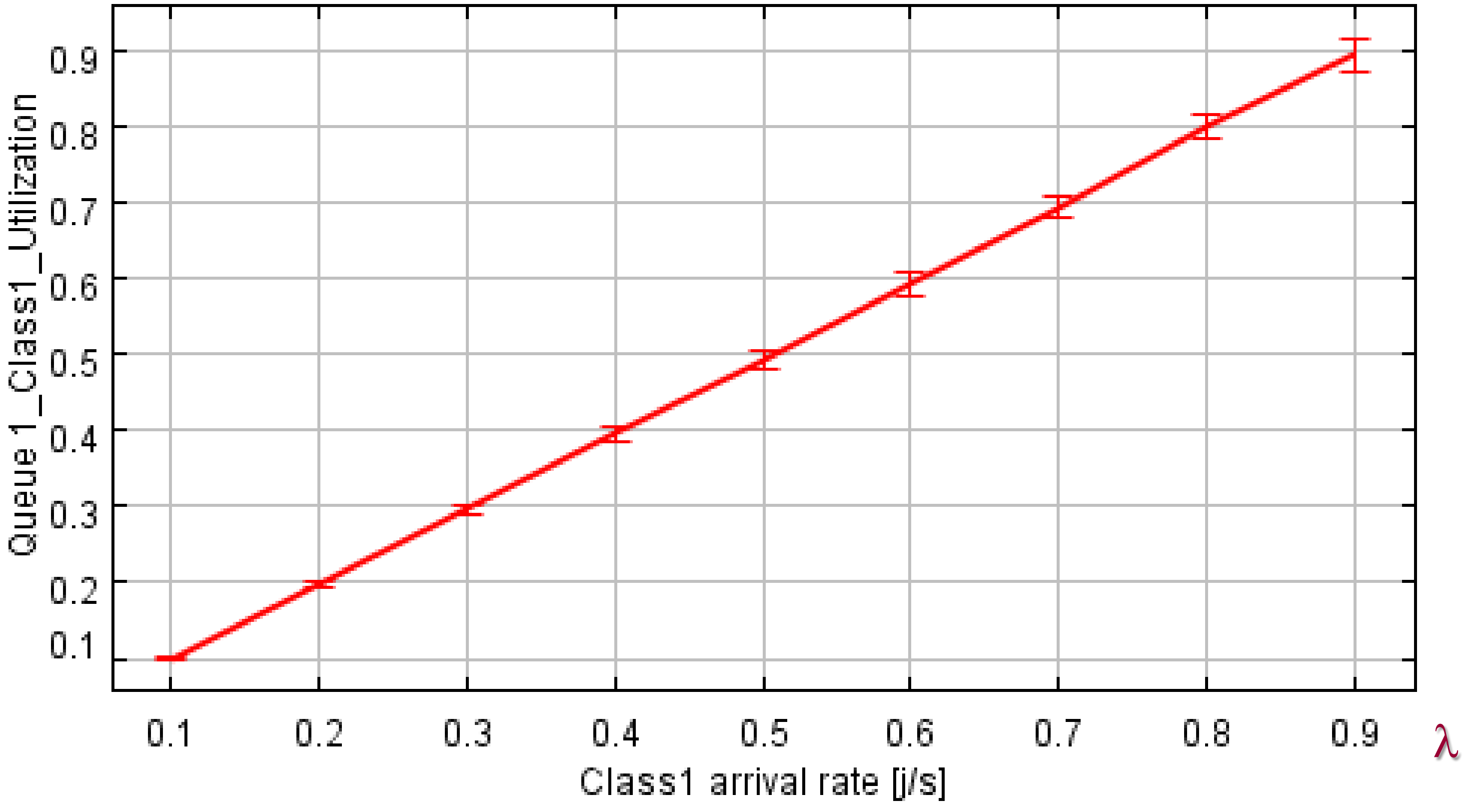
three Interarrival times distributions (with same mean)

$\lambda = 0.9$ req/sec, mean=1.11 sec

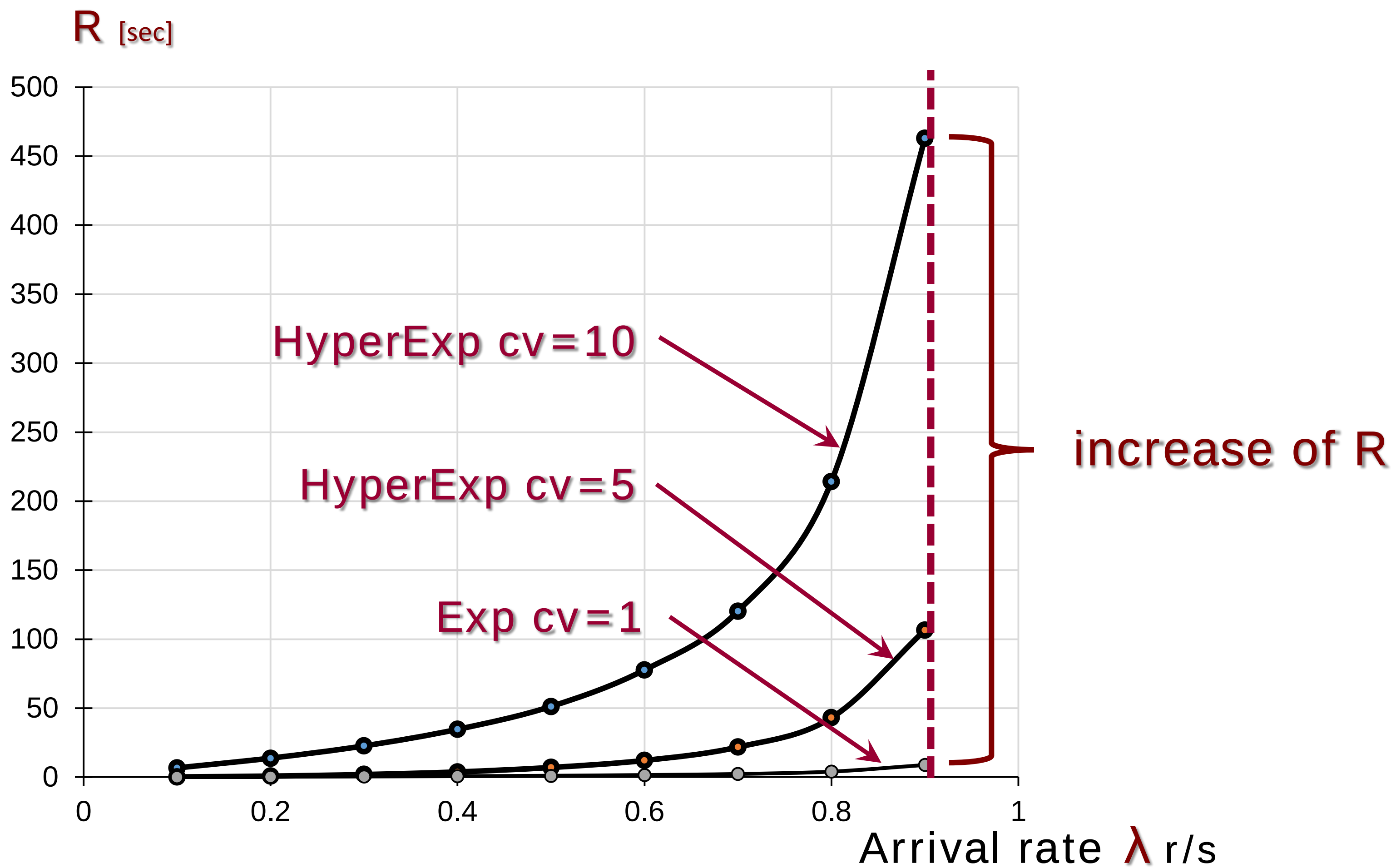


Utilization of Queue1

U = λ S **S** = 1 sec **λ** = 0.1÷ 0.9 req/sec



Response time R



R with five distributions of Interarrival times

- same Service time S=1sec and same distribution (exponential)
- five distribution of Interarrival time, coeff. of var. from 0 to 10
- with λ =0.9 req/sec --> Response time from 5.13 to 455.06 sec!

Arrival rate	Interarrival Time Distributions				
	Const c=0	Hypo c=0.5	Exp c=1	Hyper c=5	Hyper c=10
$\lambda = 0.1$ [r/s]	1.00	1.01	1.11	1.22	1.24
$\lambda = 0.3$ [r/s]	1.05	1.12	1.43	2.20	2.40
$\lambda = 0.6$ [r/s]	1.47	1.70	2.54	14.49	46.98
$\lambda = 0.9$ [r/s]	5.13	6.43	9.92	116.88	455.06

! !!!

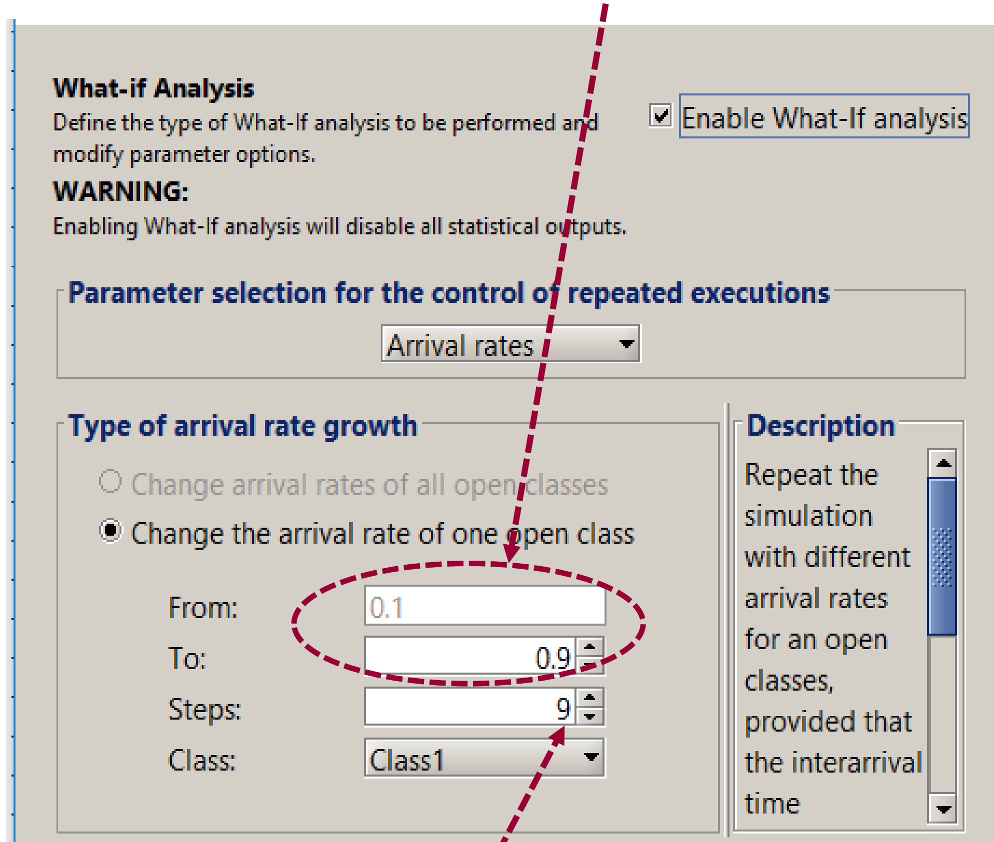
Sect.4.3 – the problem

- evaluate the impact of variability of Service times on the performance
- the system consists of a queue station
- the Interarrival times are exponentially distributed
- we consider five models with different distributions of Service times with the same mean and increasing variance, with coefficients of variation from 0 to 10
- the request arrival rate λ varies from 0.1 to 0.9 req/sec

Variability of Service times - JSIMg

Interarrival times **exp. (cv=1)** for the **five** models

initial and final arrival rates



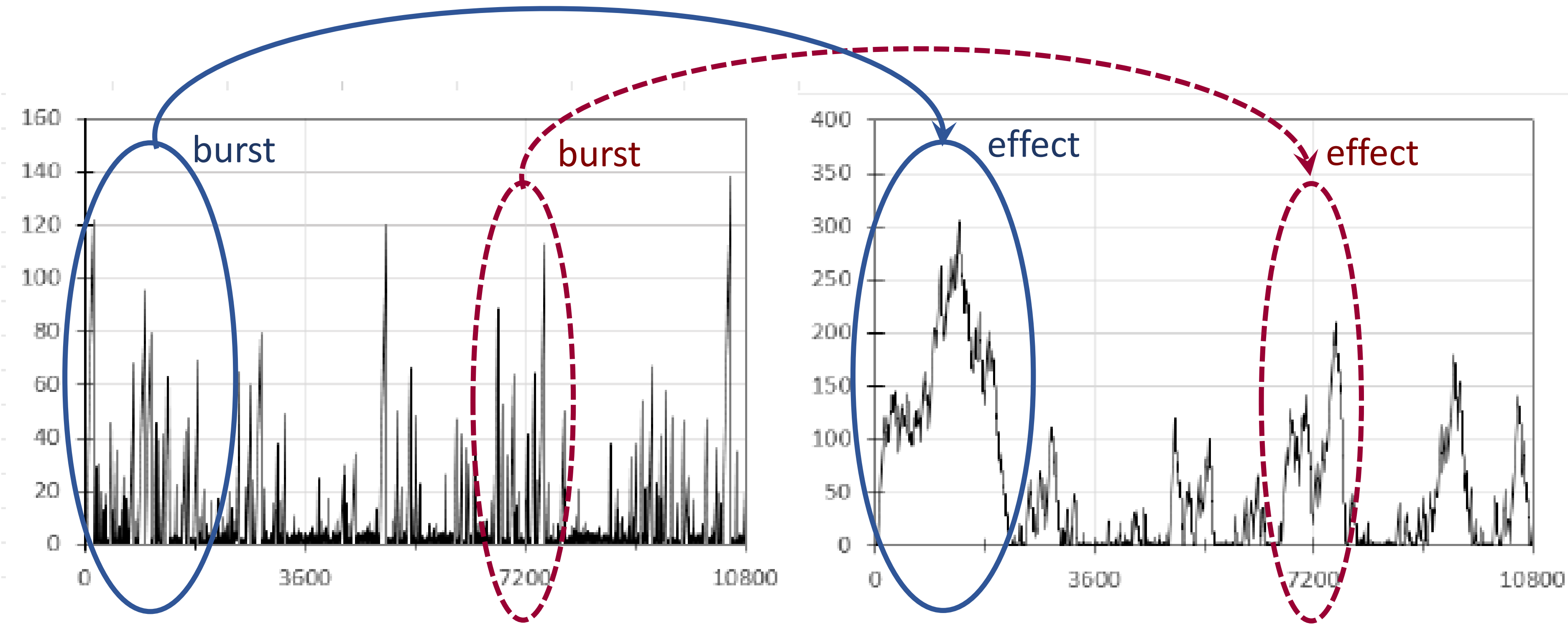
9 models requested

Service times: S=1 sec , five distrib.

- Deterministic cv=0
- Hypo-exponential cv=0.5
- Exponential cv=1
- Hyper-exponential cv=5
- Hyper-exponential cv=10

effects of bursts of Service times on Response times - JSIMg

S=1 sec   λ=0.9 req/sec   hyper-exponential   coeff.of variation=5



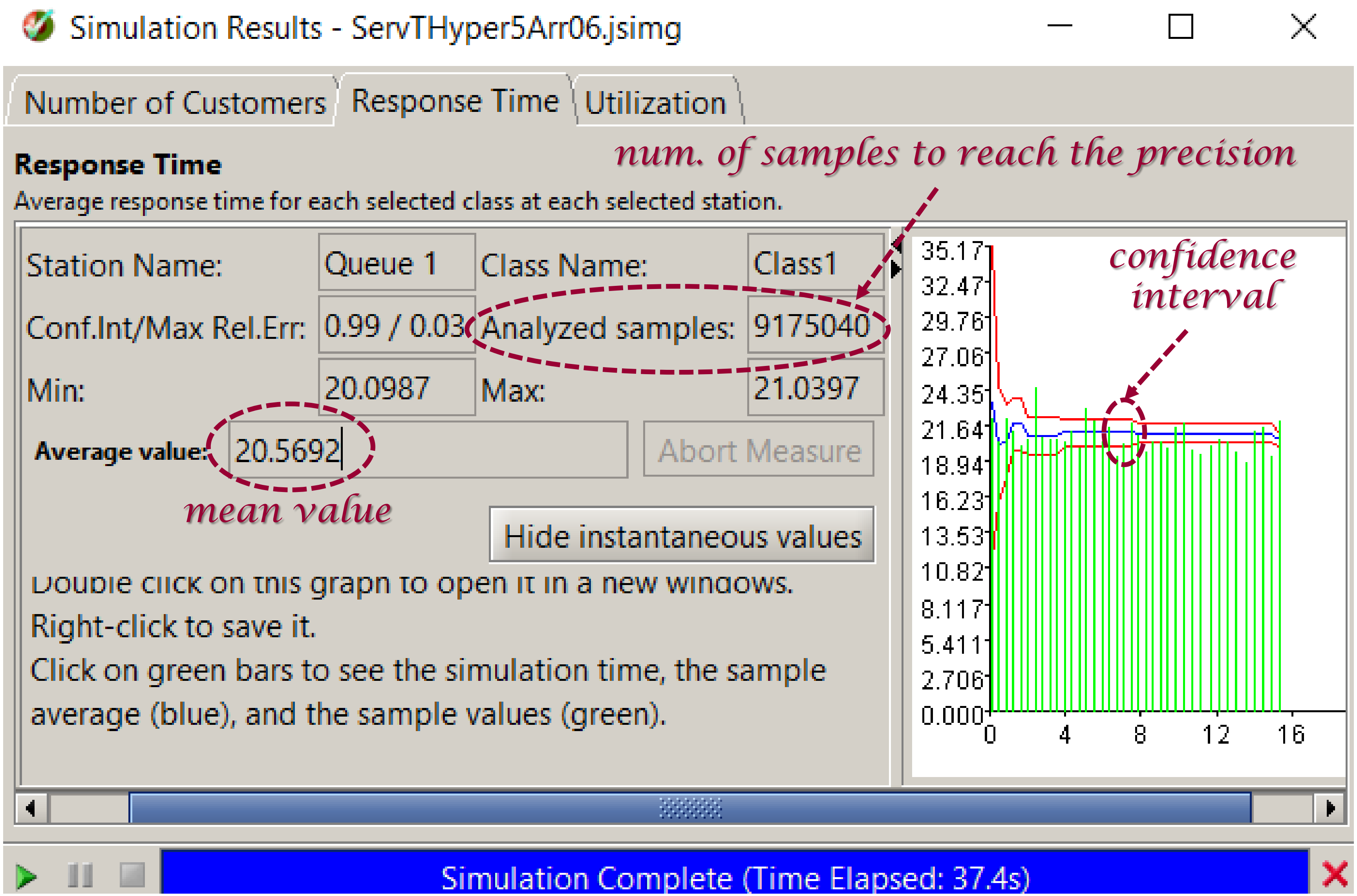
(a) Fluctuations of Hyper-exp Service Times [sec]

(b) Response times [sec]

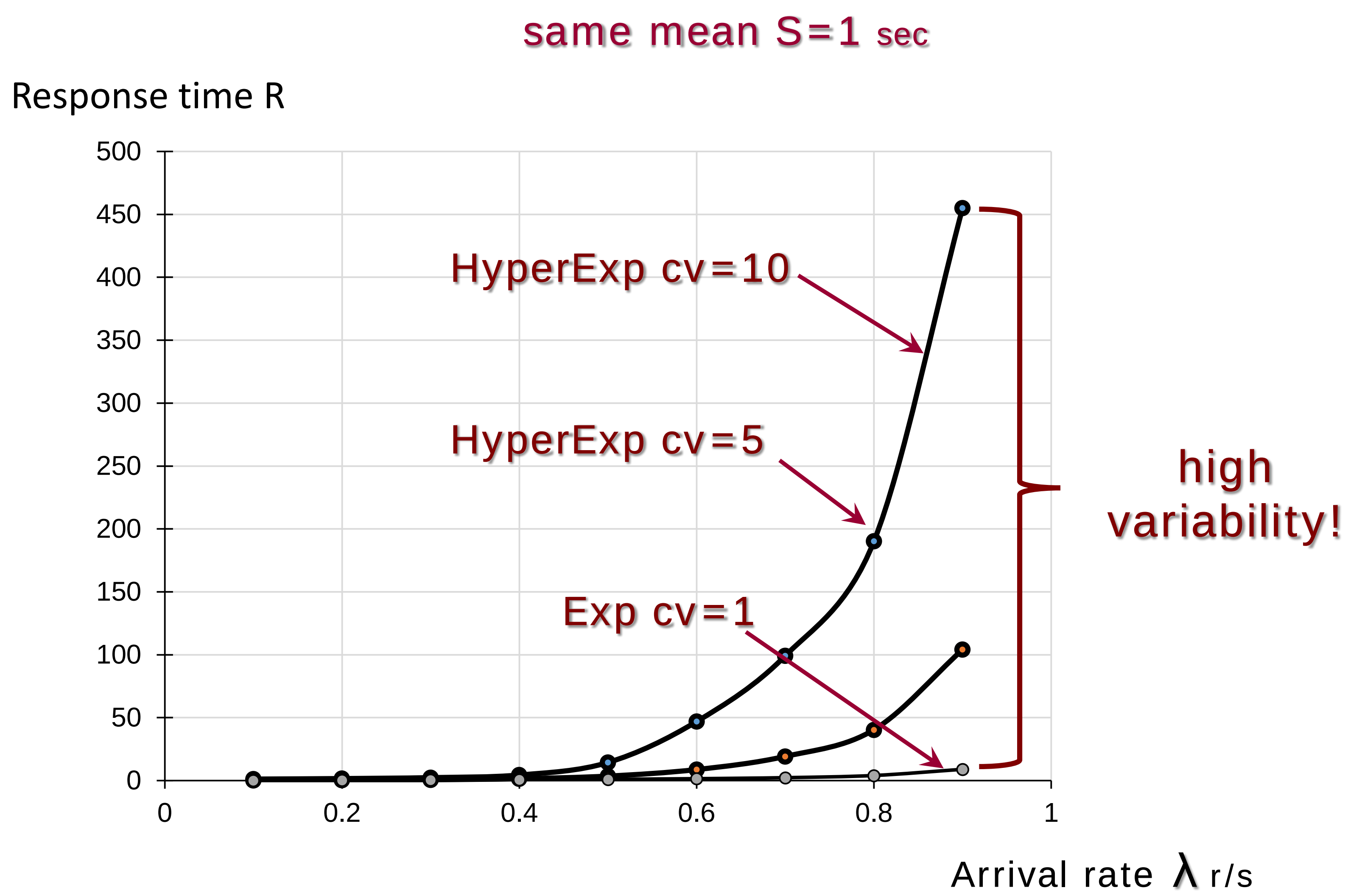
Service times S

Response times R

Response time, S=1sec Hyperexp. cv=5, λ=0.6 req/sec



Response times with three distributions of Service times



Response times R

- same Interarrival times and same exponential distribution
- five distributions of Service times, coeff. of var. from 0 to 10
- with λ =0.9 req/sec --> Response time from 5.53 to 453.36 sec!

Arrival rate	Response time				
	S=1sec Service Time Distributions				
Exp c=1	Const c=0	Hypo-exp c=0.5	Exp c=1	Hyper-exp c=5	Hyper-exp c=10
$\lambda = 0.1$ [r/s]	1.05	1.06	1.11	2.42	6.67
$\lambda = 0.3$ [r/s]	1.21	1.26	1.43	6.66	22.62
$\lambda = 0.6$ [r/s]	1.76	1.95	2.54	20.56	77.15
$\lambda = 0.9$ [r/s]	5.53	6.51	9.92	119.17	453.36
$\lambda = 0.9$ [r/s]	**5.5**	**6.625**	**10**	**118**	**455.5**

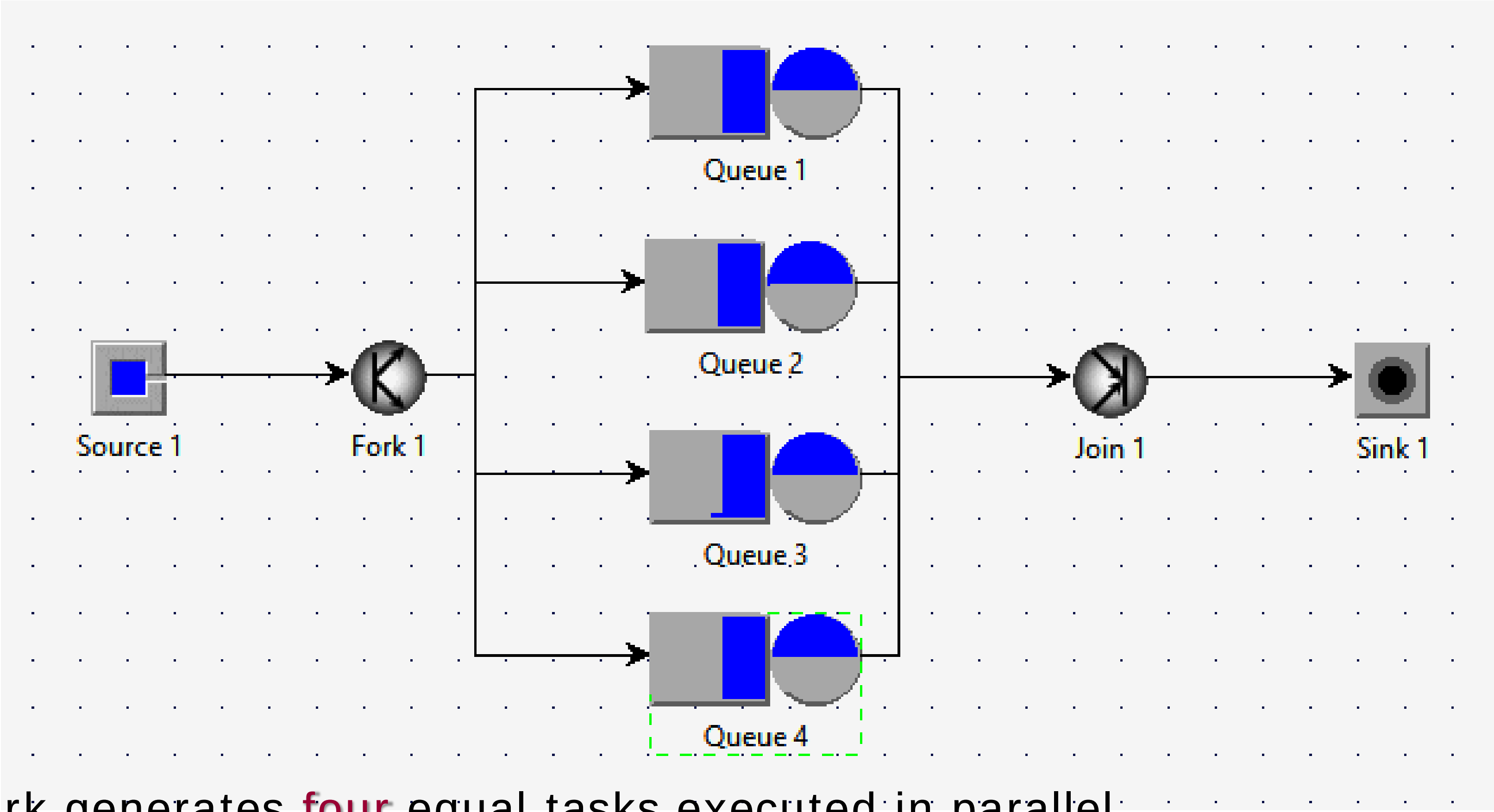
M/G/1 exact values

Parallel Computing

Chapter 5 --- Sect.5.1 - 5.2 - 5.3

open model
tool used: JSIMg

Sec.5.1 Synchronization of all parallel tasks at the Join



- Fork generates four equal tasks executed in parallel
- interarrival times of jobs and service times are exponentially distributed
- Service times with the same mean $S_i = 0.5$ s , $\lambda = 1 \div 1.8$ j/s
- Response time of the system?

System Response time

Number of Customers | Response Time | Utilization | System Throughput | System Number of Customers | Fork Join Response Time

Response Time

Response Time for each chosen F/J area

Description

Station Name: Fork 1

Class Name: Class1

Conf.Int/Max Rel.Err (0-1): 0.99 / 0.03

Samples: 5

X min: 1

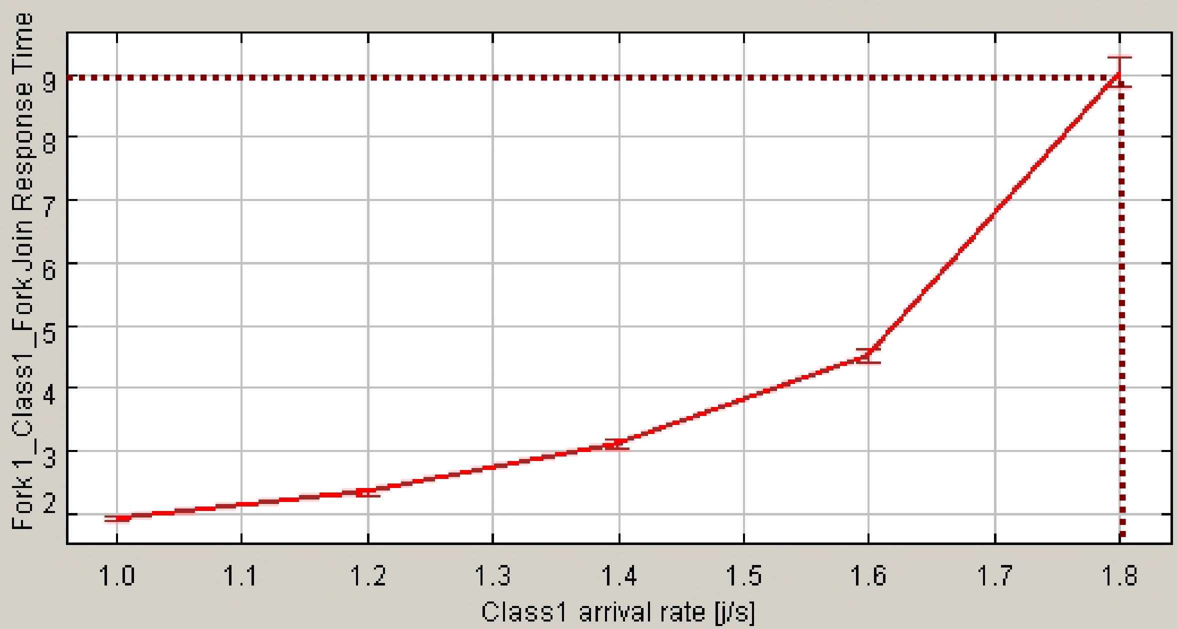
X max: 1.8

Y min: 1.875

Y max: 9.28

Plot

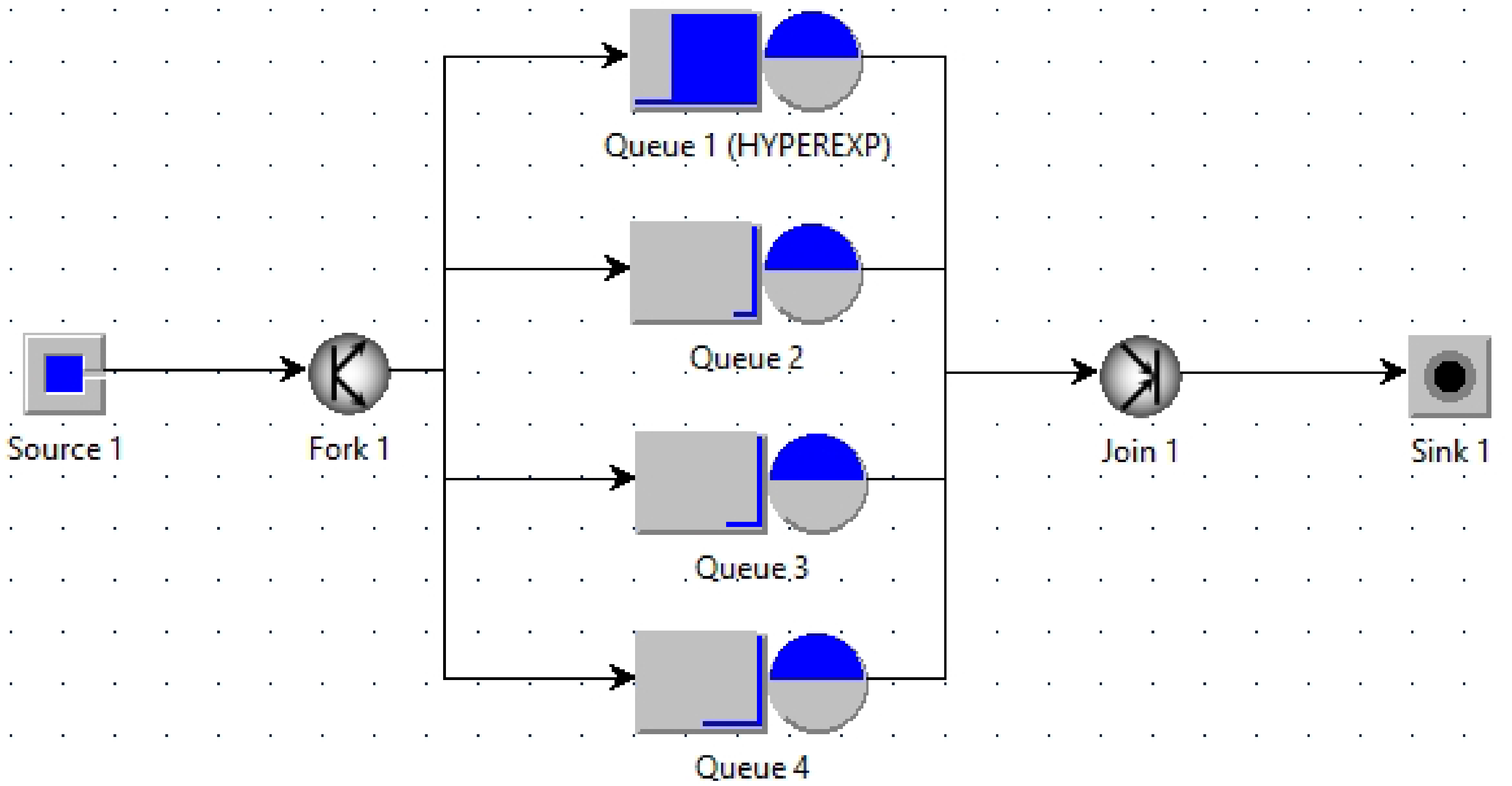
☑ Show confidence interval r



Simulation results table

*	1.000 j/s	1.200 j/s	1.400 j/s	1.600 j/s	1.800 j/s
Mean value	1.922	2.338	3.096	4.510	9.036

Sec.5.2 Synchr. of all tasks, S_{Queue1} hyperexp, cv=3



- Fork generates four equal tasks executed in parallel
- interarrival times exponentially distributed λ =1 ÷ 1.8 j/s
- Serv. times same mean 0.5s, S_{Queue1} hyperexp cv=3 - S_1,S_2,S_3 are exp
- Response time of the system?

System Response time

Number of Customers | Response Time | Utilization | System Number of Customers | System Throughput | Fork Join Response Time

Fork Join Response Time

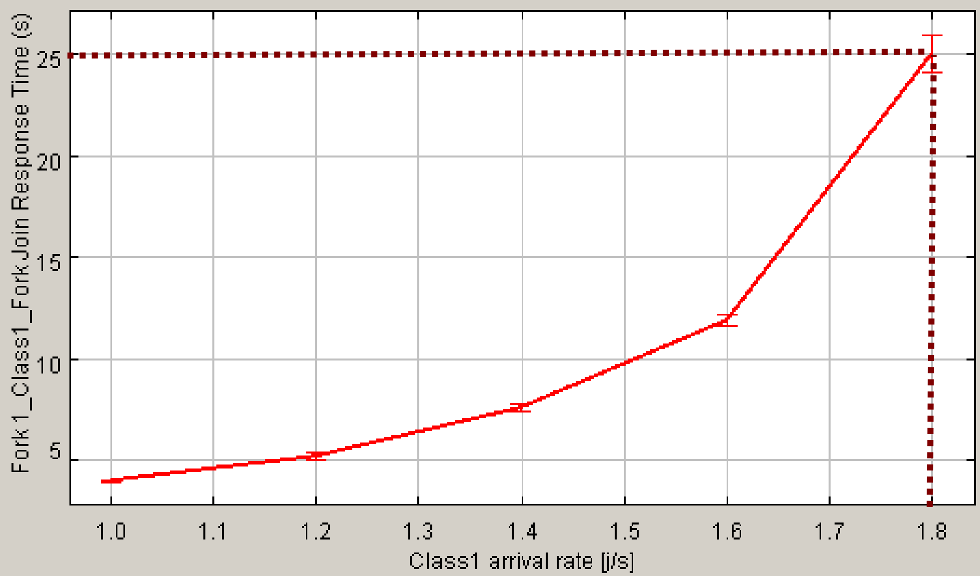
Average response time for each chosen class in each chosen Fork/Join section.

Description

Station Name:	Fork 1 (Fork Join)
Class Name:	Class1
Conf.Int. / Max Rel.Err. (0-1):	0.99 / 0.03
Models:	5
X min:	1
X max:	1.8
Y min:	3.863
Y max:	25.994

Plot

☑ Show confidence interval range



Simulation Results

*	1.000 j/s	1.200 j/s	1.400 j/s	1.600 j/s	1.800 j/s
Mean Value (s)	3.9515	5.1908	7.5642	11.8934	25.0649

Sec.5.3 Synchr. on the fastest task, Quorum=1

Number of Customers | Response Time | Utilization | System Throughput | System Number of Customers | Fork Join Response Time

Response Time

Response Time for each chosen F/J area

Description

Station Name: Fork 1

Class Name: Class1

Conf.Int/Max Rel.Err (0-1): 0.99 / 0.03

Models: 7

X min: 0.25

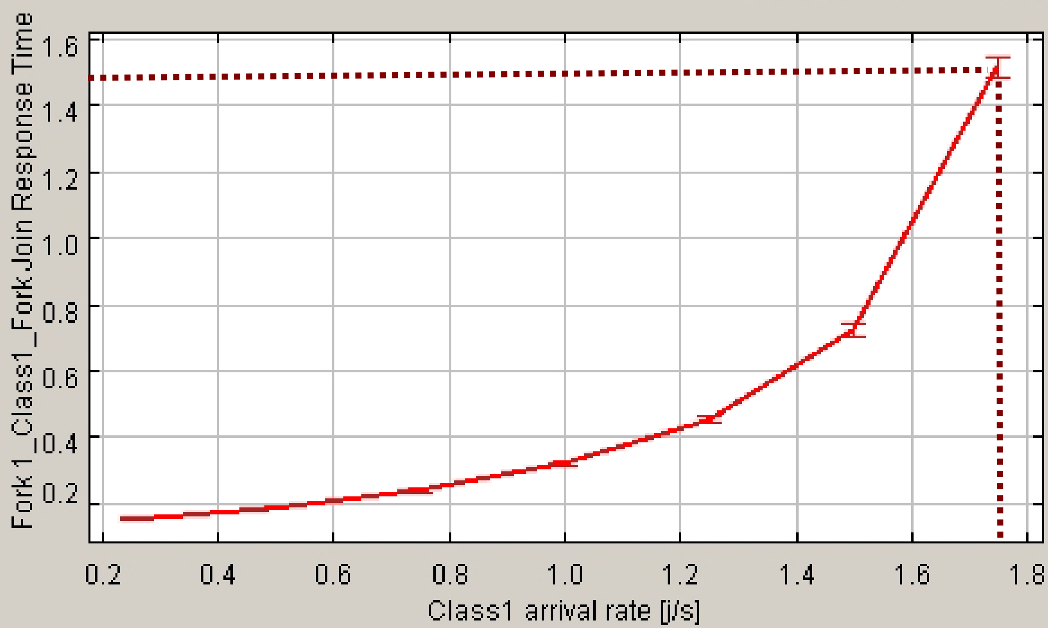
X max: 1.75

Y min: 0.15

Y max: 1.547

Plot

☑ Show confidence interval ra



Simulation results table

*	0.250 j/s	0.500 j/s	0.750 j/s	1.000 j/s	1.250 j/s	1.500 j/s	1.750 j/s
Mean value	0.152	0.188	0.239	0.320	0.452	0.722	1.516

A Facial Recognition Surveillance System

(Edge computing)

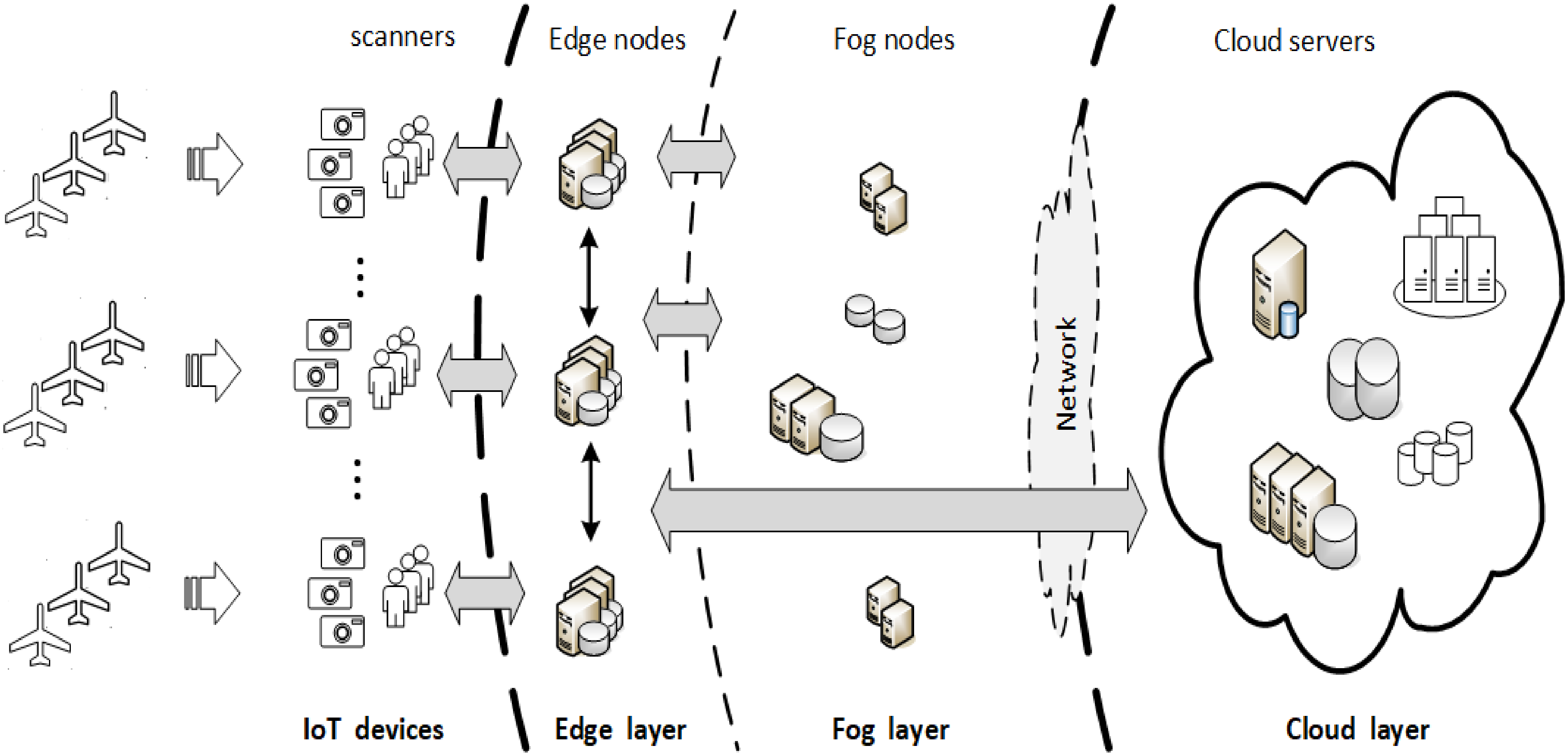
Chapter 6 --- Sect.6.1

open model
two class workload
tool used: JSIMg

Sect. 6.1 - the surveillance system architecture

- identification of people flowing in an airport detecting the faces of persons passing by the scanners
- scanners are connected to the nearest Edge nodes, that are controlled by Fog servers to activate the reaction actions
- five types of persons (scans categories): regular, suspect, dangerous, unknown person, poor-quality image
- all scan categories but the unknown are processed by the Edge nodes (require only access to the in-memory db of face images of each node)
- scans of unknown category are sent to a remote cloud equipped with a very large NoSQL db of face images (with biometrics data)
- each Edge node is initially configured with a rack and a server
- as the load increases, the number of servers will increase

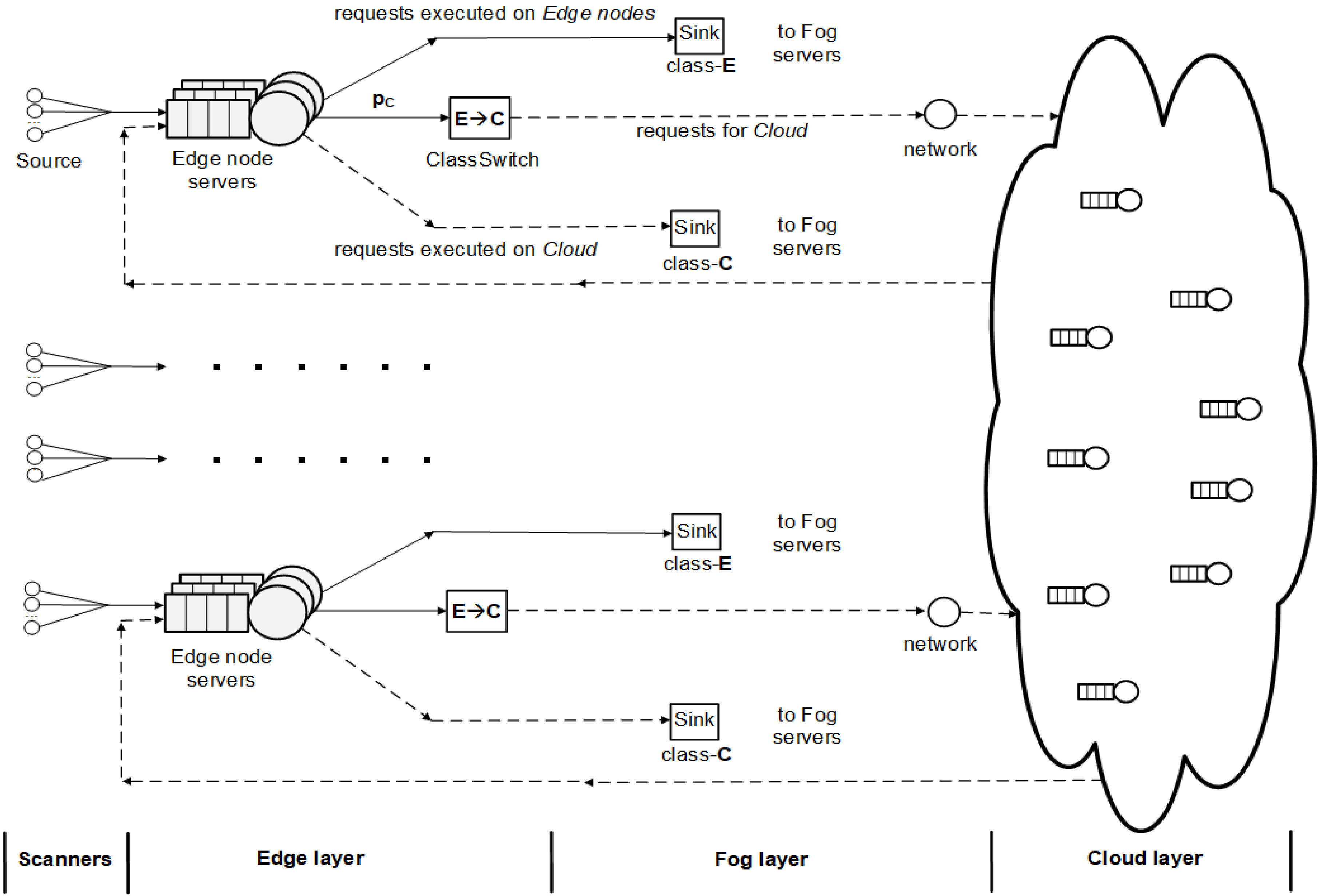
the facial identification system in the airport



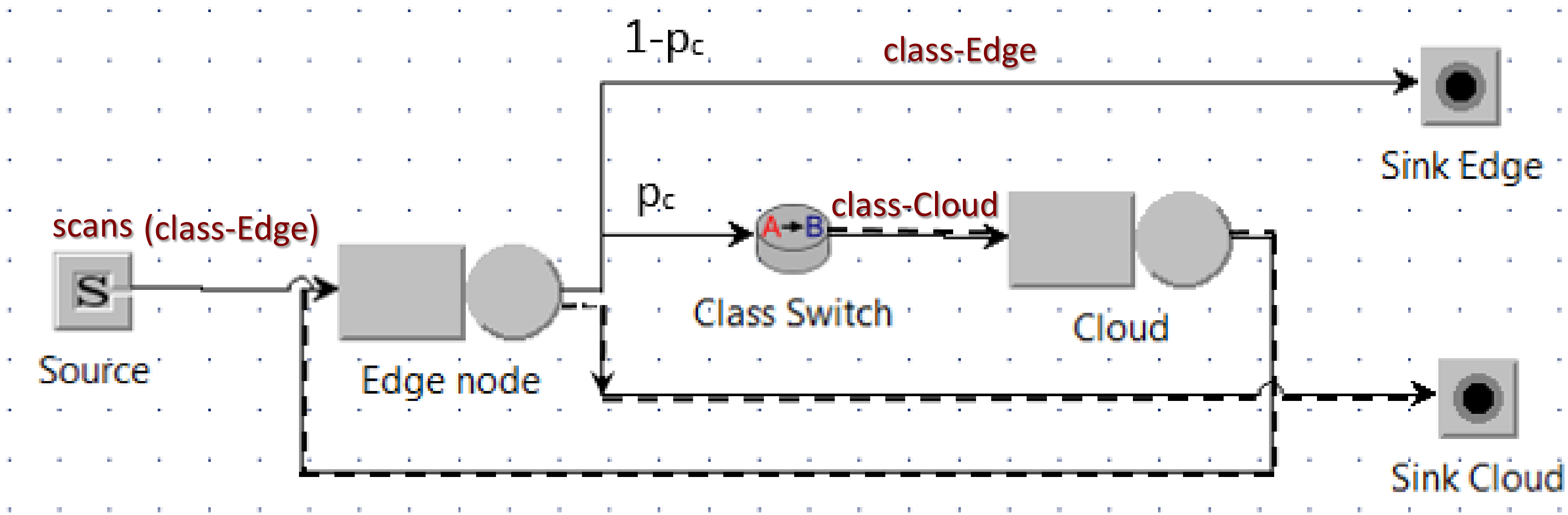
some objectives of the study

- the system must be autoscaling: it automatically increases the capacity of the Edge nodes to meet the time constraint for a facial recognition
- continuously monitors the response times of each Edge node (analysis time of a scan) and adds new servers when the performance target is approached
- Performance Constraint: the mean analysis time of the scans (except those of unknown type) required by an Edge node must be <= 3sec (threshold value), time required by the reaction actions to be effective
- Scaling Policy: when a threshold value of the recognition time of a node is approaching, a new server will be allocated on its rack (or switched to on-line status if it is already mounted)
- arriving requests to each Edge node are balanced between the servers allocated on its rack
- Fog nodes (system coordinators) are not considered in the model

model of the global system



model of an Edge node



two class of requests: Cloud, Edge

-----> *Cloud: scans of 'unknown' category* processed by Cloud servers

⟶ *Edge: other types of scans* processed by Edge node

p_c fraction of unknown scans processed by Cloud servers

Service demands of the scans (sec)

other types of scans
processed by Edge node

'unknown' scans
processed by Cloud servers

Resource (Station)	*Two classes*	
	Edge	Cloud
Edge node	0.5	0.1 p_c
Cloud	—	0.8 p_c

p_c fraction of unknown scans processed by Cloud servers

Edge node Response time, 1 server, 40% are ‘unknown’ scans

Description

Station Name: Edge node

Class Name: Edge

Conf.Int. / Max Rel.Err. (0-1): 0.99 / 0.03

Models: 32

X min: 0.2

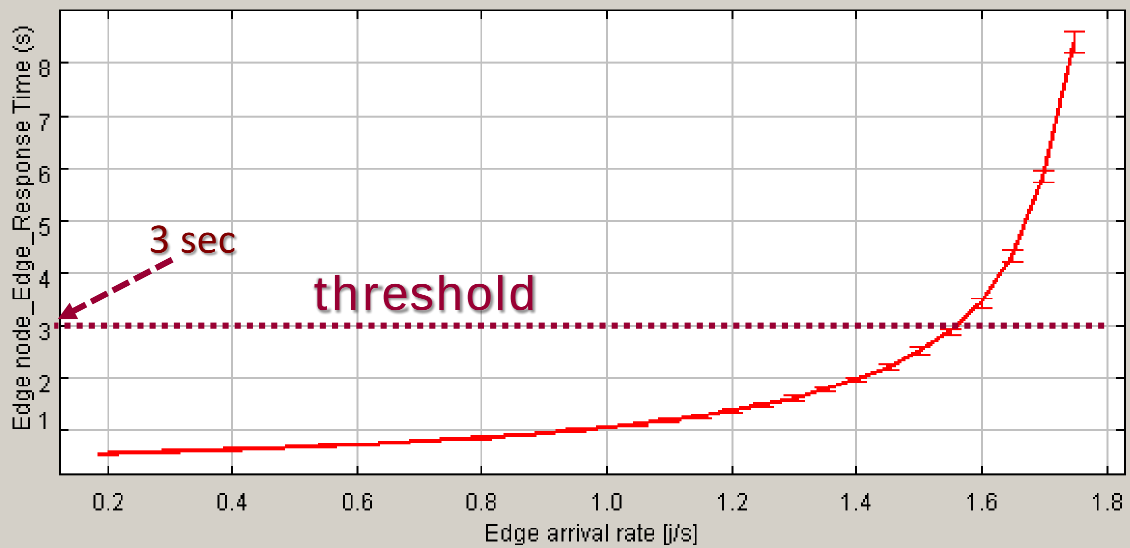
X max: 1.75

Y min: 0.543

Y max: 8.59

Plot

☑ Show confidence interval range



Simulation Results

*	1.400 j/s	1.450 j/s	1.500 j/s	1.550 j/s	1.600 j/s	1.650 j/s	1.700 j/s	1.750 j/s
Mean Value (s)	1.9608	2.1950	2.5235	2.8782	3.4366	4.3147	5.8309	8.3949
Max (s) (Conf.Int.)	2.0016	2.2503	2.5926	2.9393	3.5307	4.4214	5.9455	8.5896
Min (s) (Conf.Int.)	1.9200	2.1398	2.4544	2.8170	3.3424	4.2079	5.7163	8.2001

Response time Cloud requests

Response Time per Sink

Average response time for each selected class at each selected Sink station.

Description

Station Name: Sink 2

Class Name: Cloud

Conf.Int. / Max Rel.Err. (0-1): 0.99 / 0.03

Models: 24

X min: 0.05

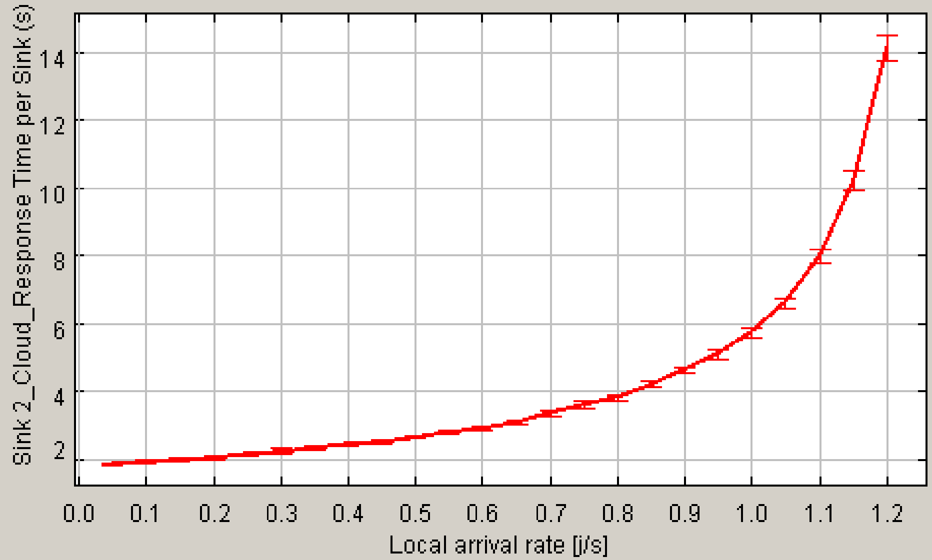
X max: 1.2

Y min: 1.821

Y max: 14.48

Plot

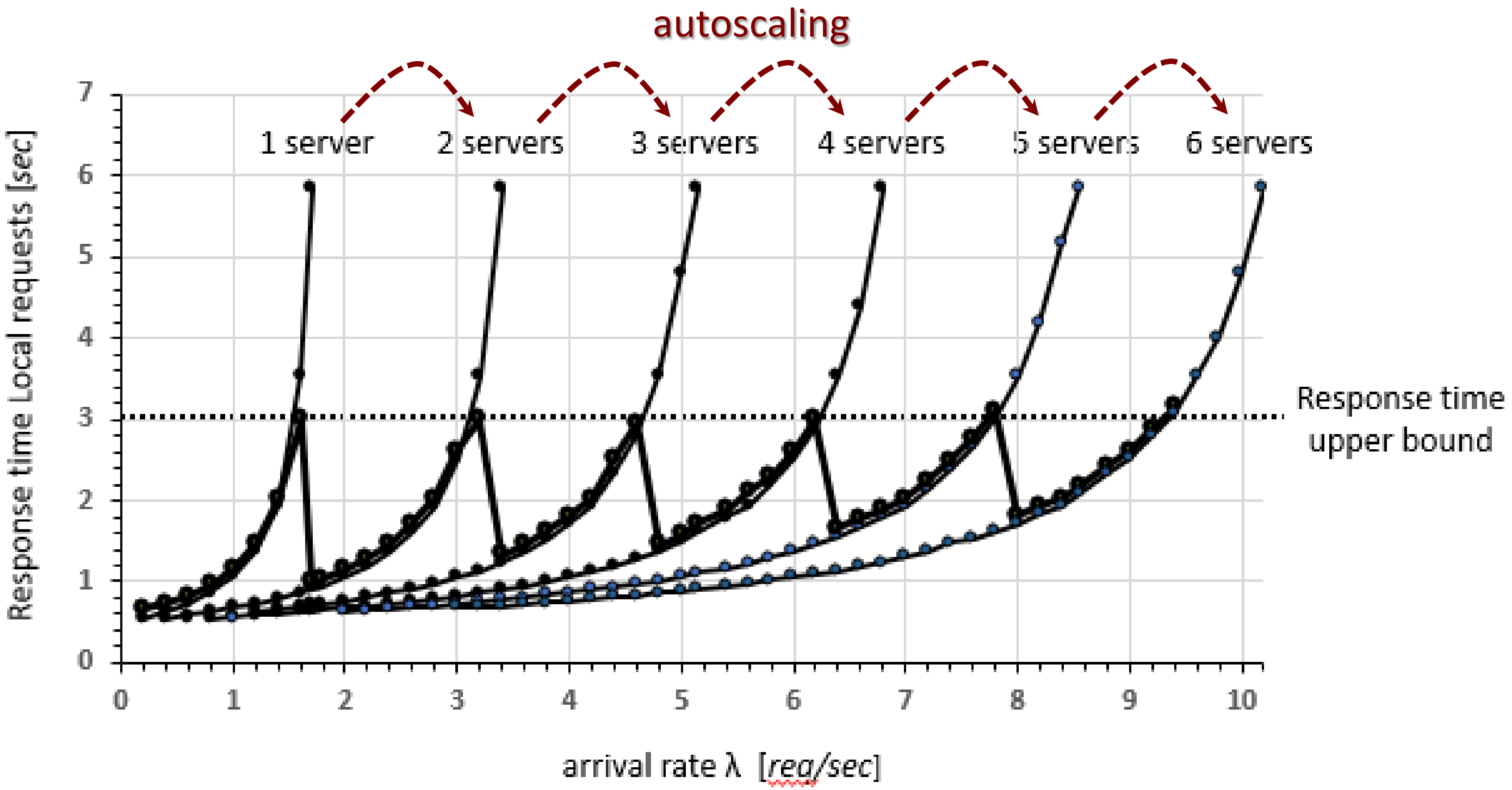
☑ Show confidence interval range



Simulation Results

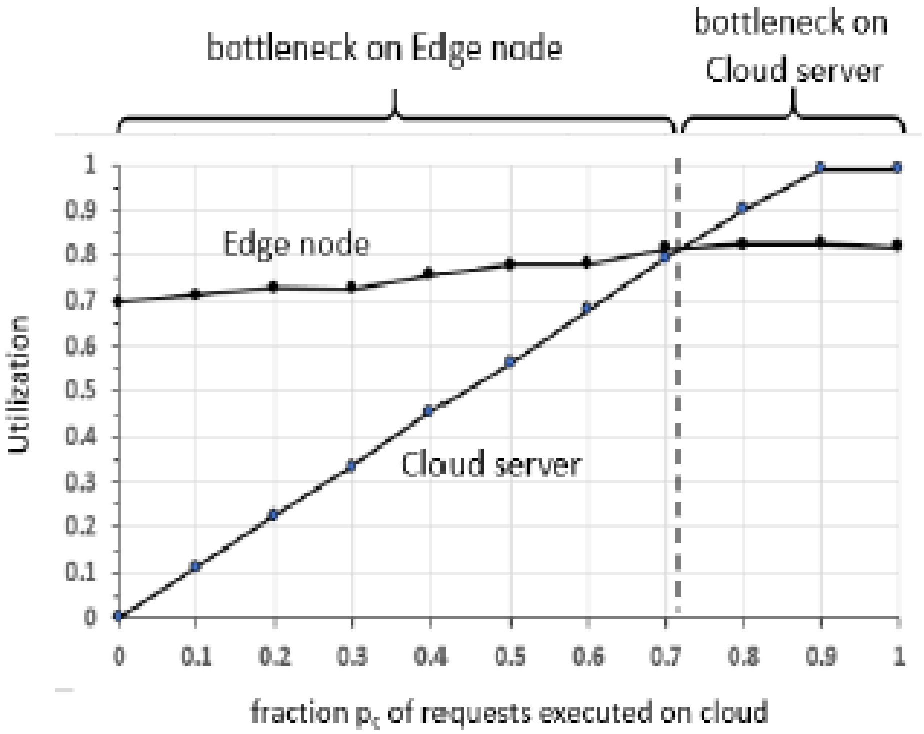
*	0.050 j/s	0.100 j/s	0.150 j/s	0.200 j/s	0.250 j/s	0.300 j/s	0.350 j/s
Mean Value (s)	1.8503	1.9156	1.9814	2.0421	2.1346	2.2251	2.3178

autoscaling effects on the Edge node Response time
(40% are 'unknown' scans)

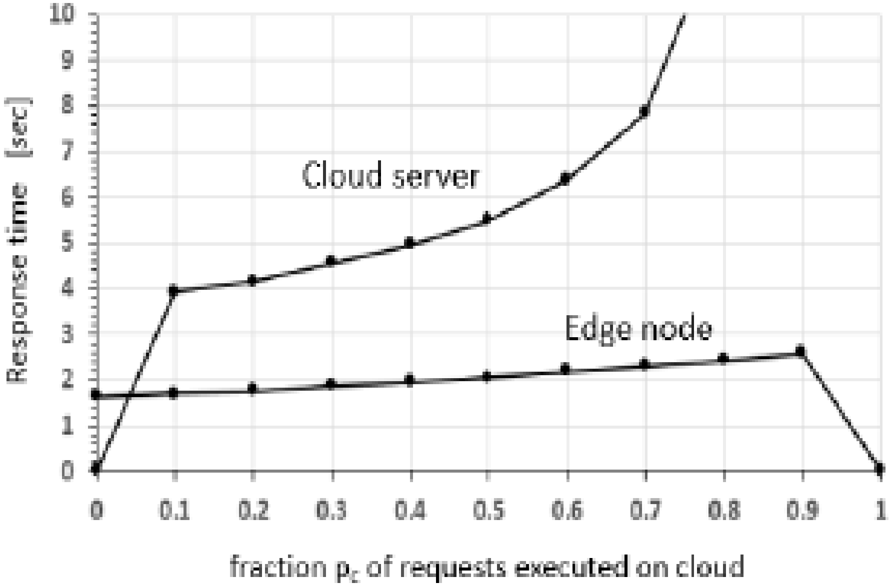


Utilizations & Response times *vs* mix of requests (λ=1.4 r/s)

Utilizations

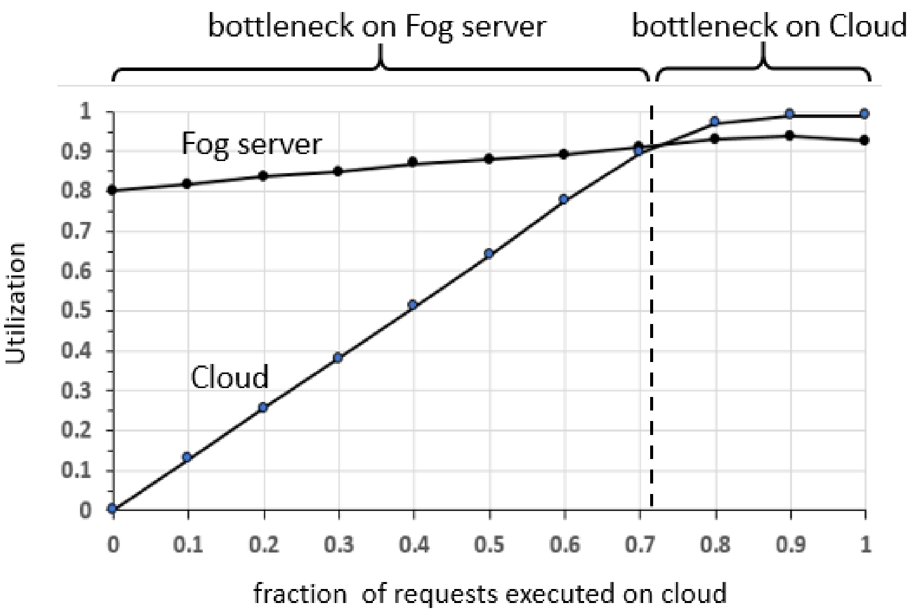


Response times

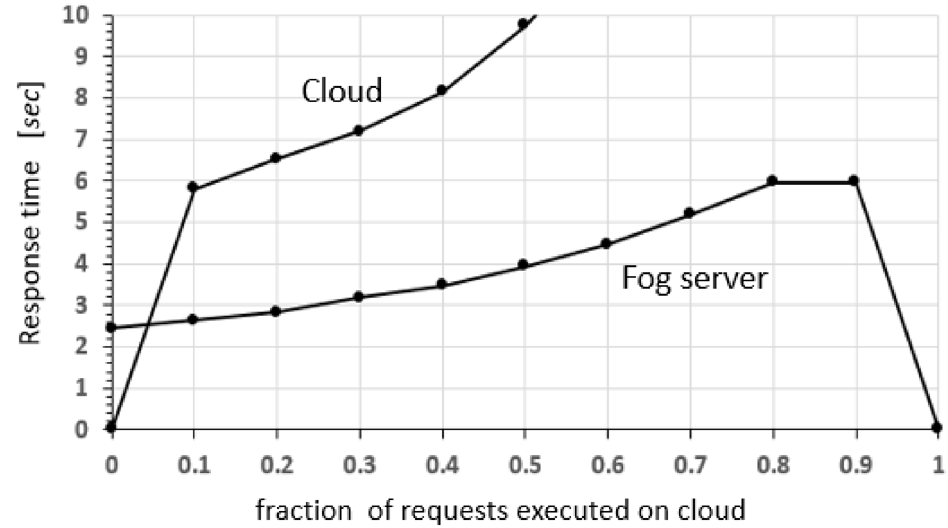


Utilizations & Response times *vs* mix of requests (λ=1.6 r/s)

Utilizations



Response times



p_c (fraction of unknown scans)

Autoscaling Load Fluctuations

Chapter 6 --- Sect.6.2

mixed model (open and closed)
two class workload
tool used: JSIMg (Queue Net+Petri Net)

Sect.6.2 - traffic spikes and variability of Service demands

- load fluctuations on the servers of private and public data centers of service providers are due to the combined effects of the variability of the incoming traffic rate and computation time of service requests
- fluctuations have very different intensities, durations and time scales
- long-term fluctuations: low frequency, small/medium intensity, generated by the typical growth trend of workloads
- short-term fluctuations: short duration, high frequency, high intensity, can occur at unpredictable instants of time

- the right sizing problem: minimum number of resources that must be used as much as possible to achieve the performance objectives
  - over-provisioning -> wastage of resources and money
  - under-provisioning->violation of customer expectations in terms of SLA

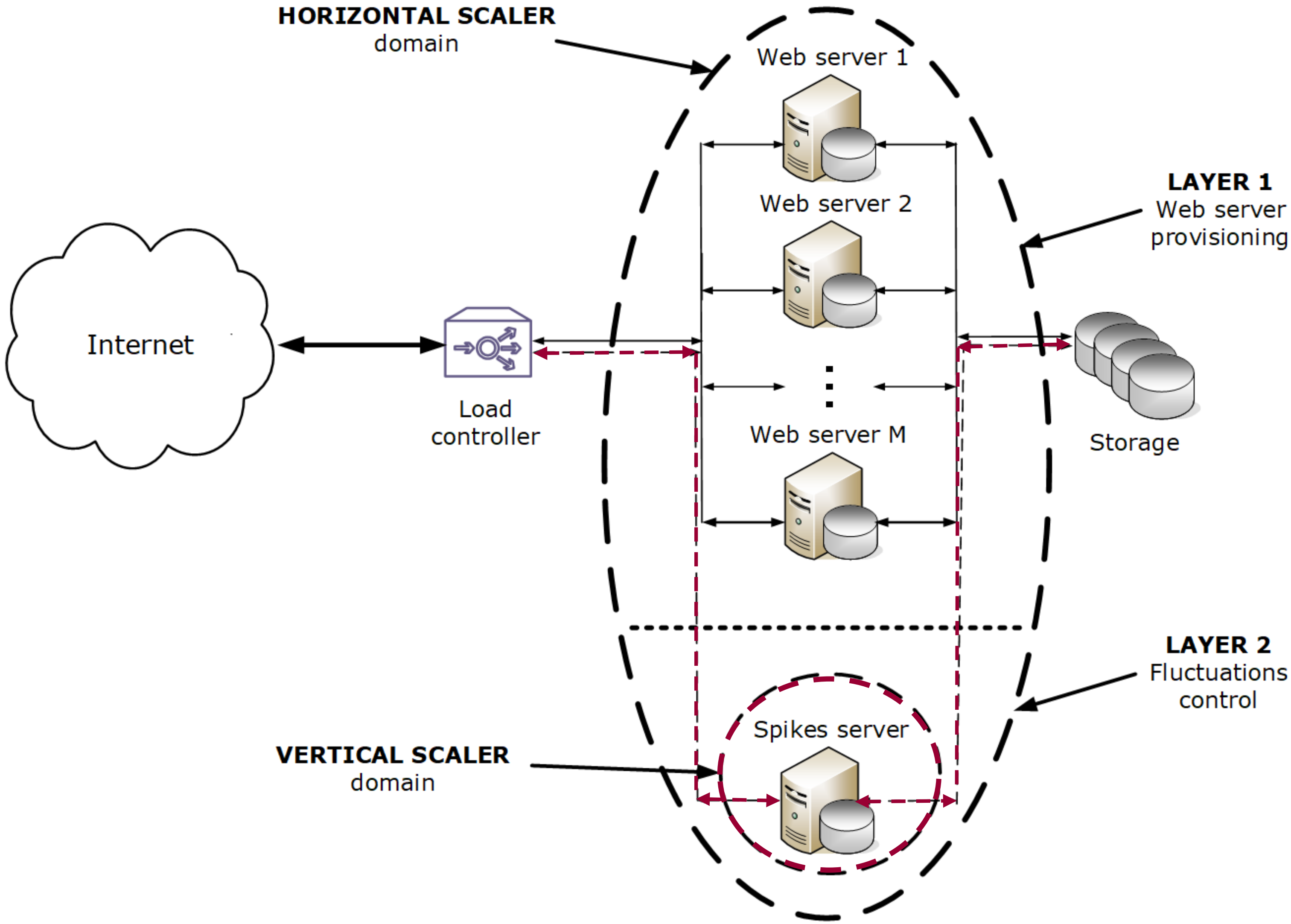
horizontal autoscaling

- dynamic provision/deprovision of resources that should be used as much as possible to achieve the performance target with minimal cost (e.g., AWS Auto Scaling, Microsoft Azure autoscale, ...);
- good results with workloads subject to long-term fluctuations, typically generated by physiological trends of the load (growth rate that increases progressively and continuously)
- BUT ... with workloads that have short-term fluctuations very often the results are quite unsatisfactory!

WHY?

- the presence of load fluctuations that typically have a high rate of occurrence and peak of values of short duration
  - has a negative impact on performance (dynamic resource congestions are responsible for very high response times)
  - can foster contradictory scaling decisions which, in short time intervals, generate dangerous oscillations in the number of resources provisioned

the hierarchical autoscaler scenario



hierarchical horizontal scaling with two layers

- enhance the horizontal scaler (Layer 1) with a second layer consisting of a Spike Server for the execution of load peaks

- Layer 2: activated when a high-load state (which usually precede a load peak) is detected in a Web Server, new arriving requests are automatically routed to the Spike Server

- new performance metric monitored by the autoscaler, the Spike Indicator (SI): number of requests in execution in a Web Server, metric that capture the fluctuations in both the arriving traffic and service demands

- the alarm threshold SI^{max} is set and is used as a autoscaler metric-based rule to activate traffic routing towards Spike Server

- a single Spike Server may execute the peak loads of several Web Servers

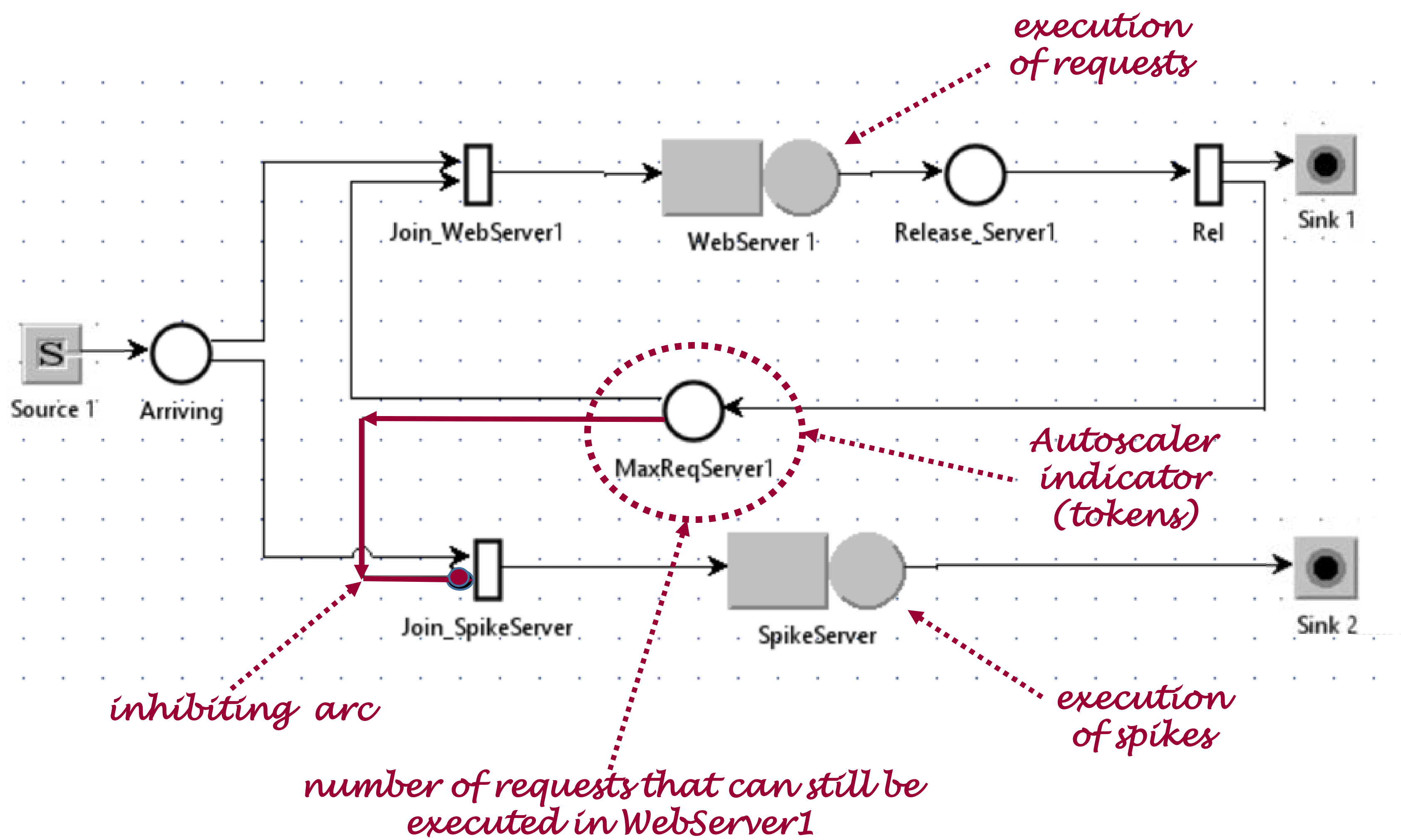
the implemented model

- is focused on evaluating of the impact on data center performance of the dynamic routing of peak loads to the Spike Server
- we do not model Layer1 actions (with only one Web Server) for the resource provisioning but we are concentrated on Layer 2 actions
- the performance indicator triggered at Layer 1 for the resource provisioning is System Response Time R_0 (the mean of the response times of Web Server and Spike Server for the arriving requests)
- a target value of R_0 is set

⇓

- we study the behavior of system performance with respect to
  - arrival rate of requests 1 ÷ 12 req/sec
  - alarm threshold values SI^{max} of Spike Indicator 10 ÷ 160 req
  - vertical scaling of CPU share of Spike Server, from 40% to 80%

autoscaler model with one WebServer1 and the SpikeServer



operational steps of the scaler

- Layer 1: monitor the metric System response time R_0 (of the arriving requests) and make scaling decisions concerning the provisioning of new servers when its target value is reached

- Layer 2: control of load fluctuations by monitoring the Spike Indicator in WebServer1; when Spike Indicator SI> alarm threshold -> activate dynamic routing of new arriving requests to SpikeServer

- when SI falls below its alarm threshold, new incoming requests will be routed again to WebServer1

- if, despite the above actions the System response time approaches its target value, it can be further decreased by vertical scaling the SpikeServer increasing the CPU share (if available) for the Web app

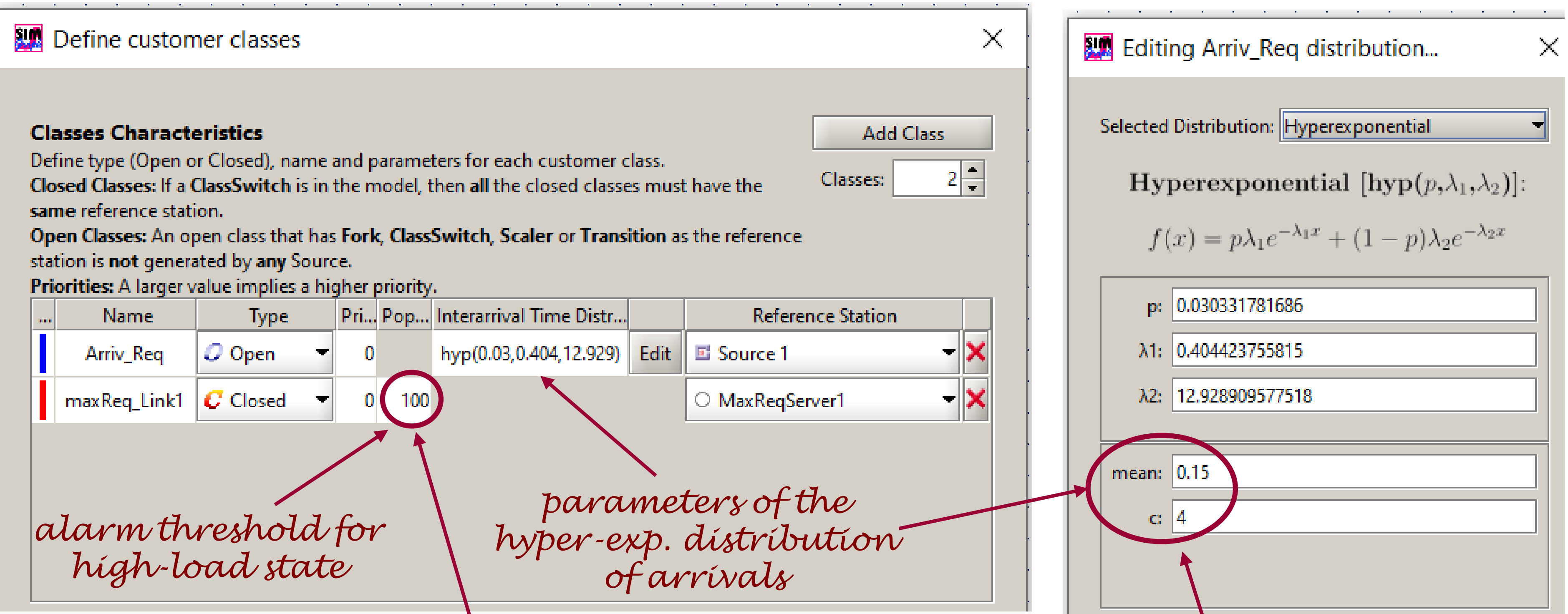
- if System response time does not drop below the target value with the above actions, a new WebServer must be provisioned at Layer 1

workload

- two classes of customers: arriving requests (open), tokens (closed)
- tokens (closed class): number of requests that are in execution in WebServer1 (their maximum value is SI^{max}), initially all SI^{max} are located in *place* MaxReqServer1

- to reproduce the fluctuations:
  - in arriving requests (open class): distribution of interarrival times: hyper-exponential, from 1 to 11 req/sec, *coeff. of variation c=4*
  - in service demands of the servers: hyper-exponential, mean=160 ms, *coeff. of variation c=4*

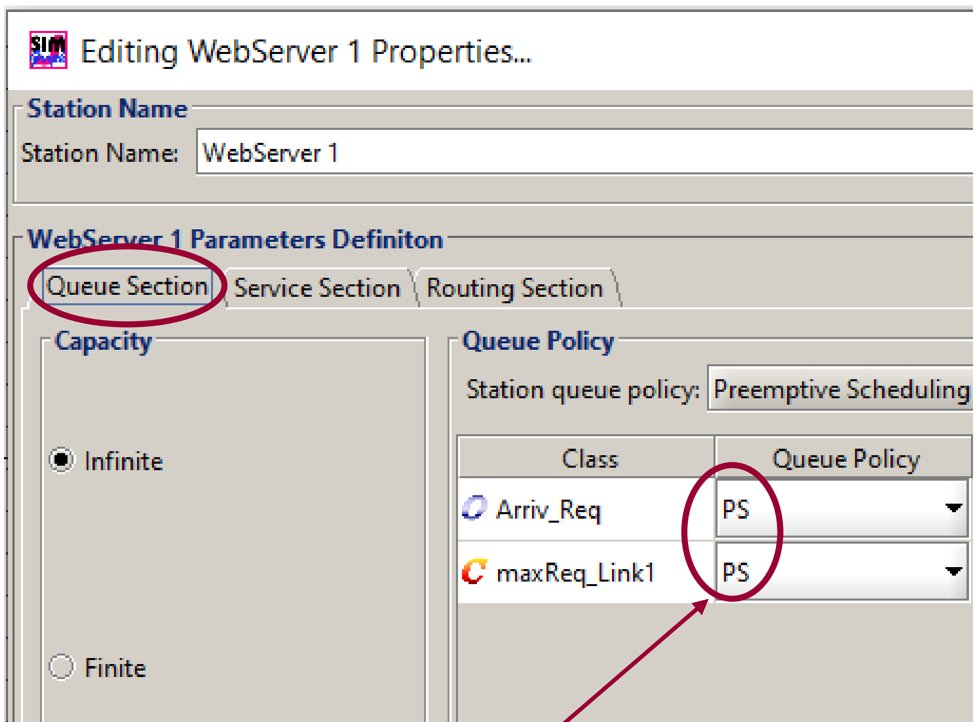
workload parameters (2 class of customers)

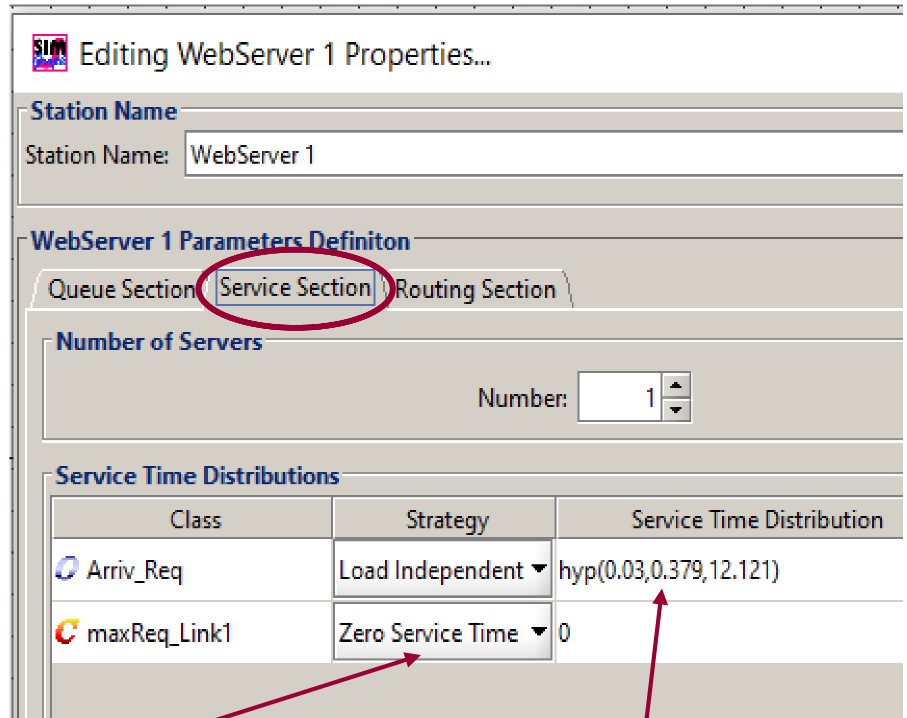


max number SImax of requests that can be in execution in WebServer1

arrival rate 6.66 req/s

parameters of WebServer1 queue station



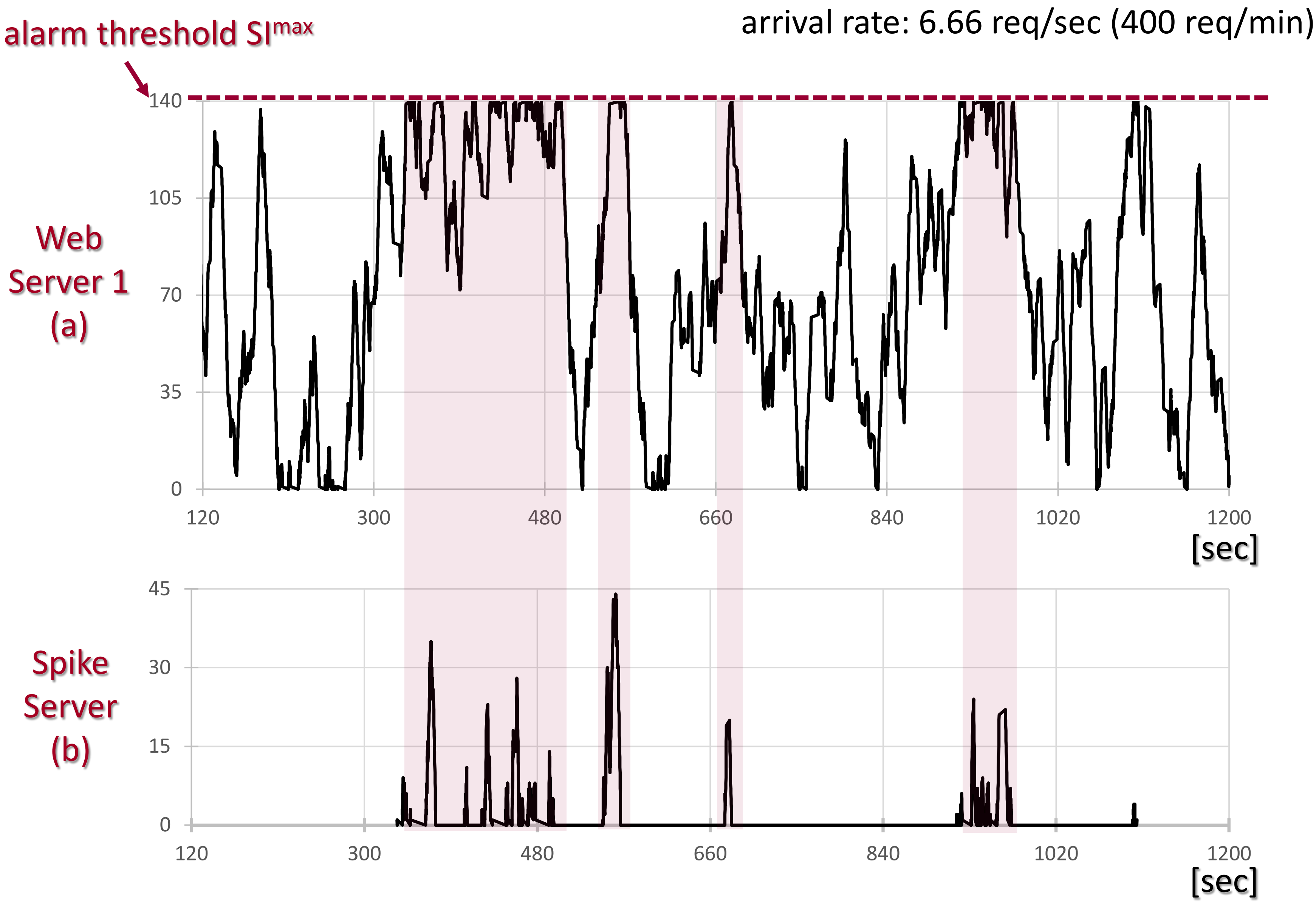


Processor Sharing scheduling algorithm

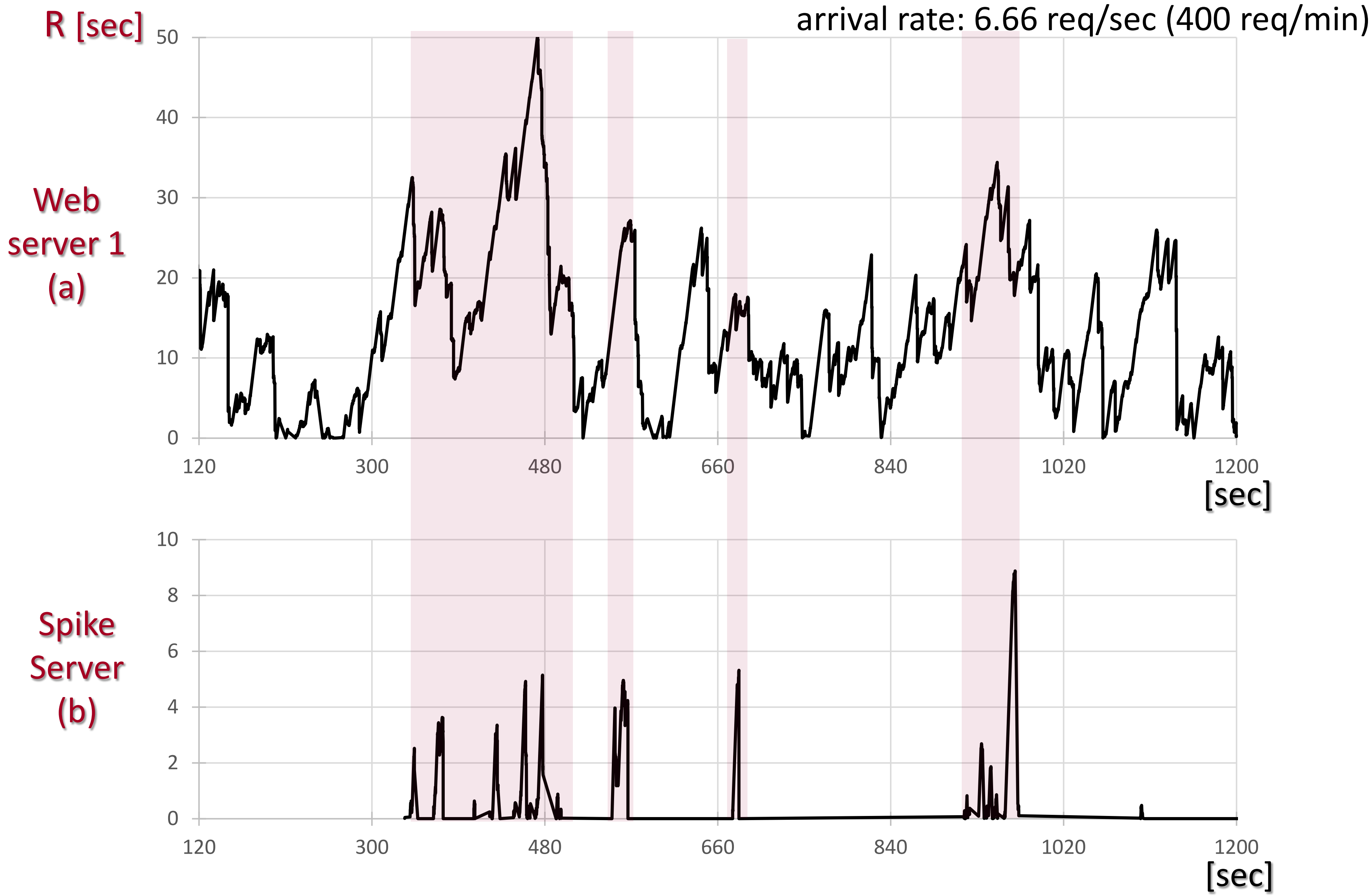
Zero service time for the tokens

hyperexp distrib. for service demands of arriving requests

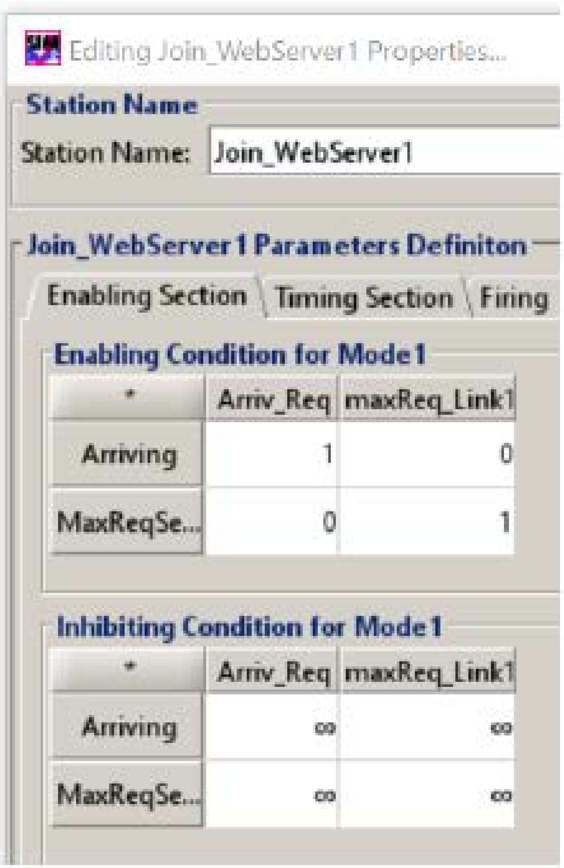
the model dynamic behavior: number of requests in exec.



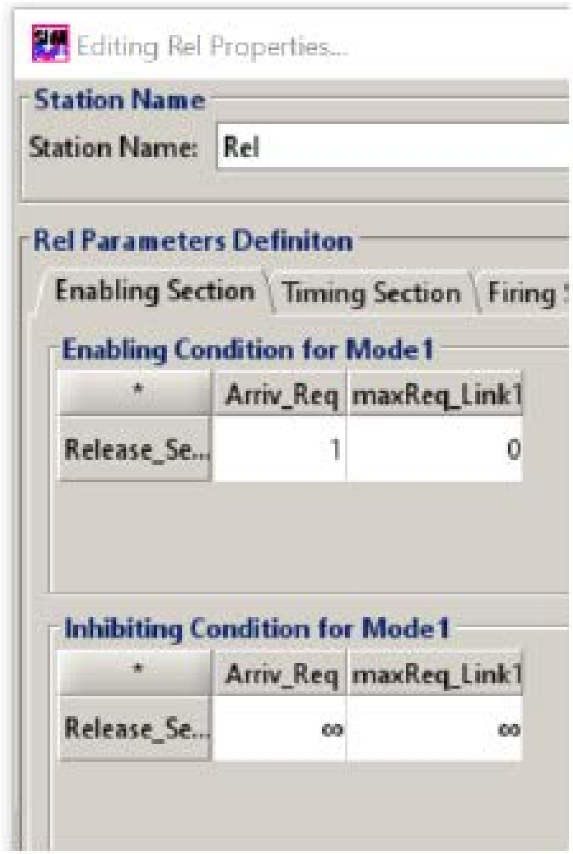
the model dynamic behavior: Response times of two servers



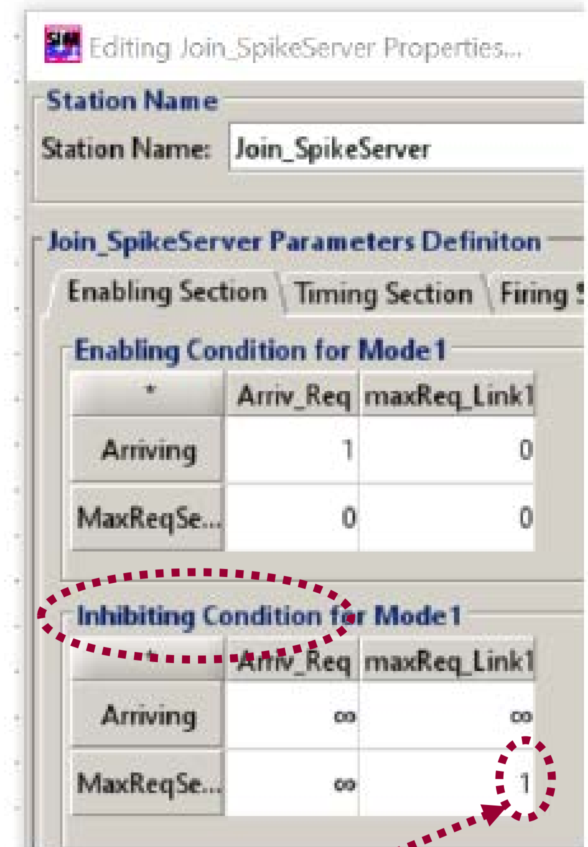
Enabling/Inhibiting conditions of the three transitions



(a) Transition JoinWebServer1



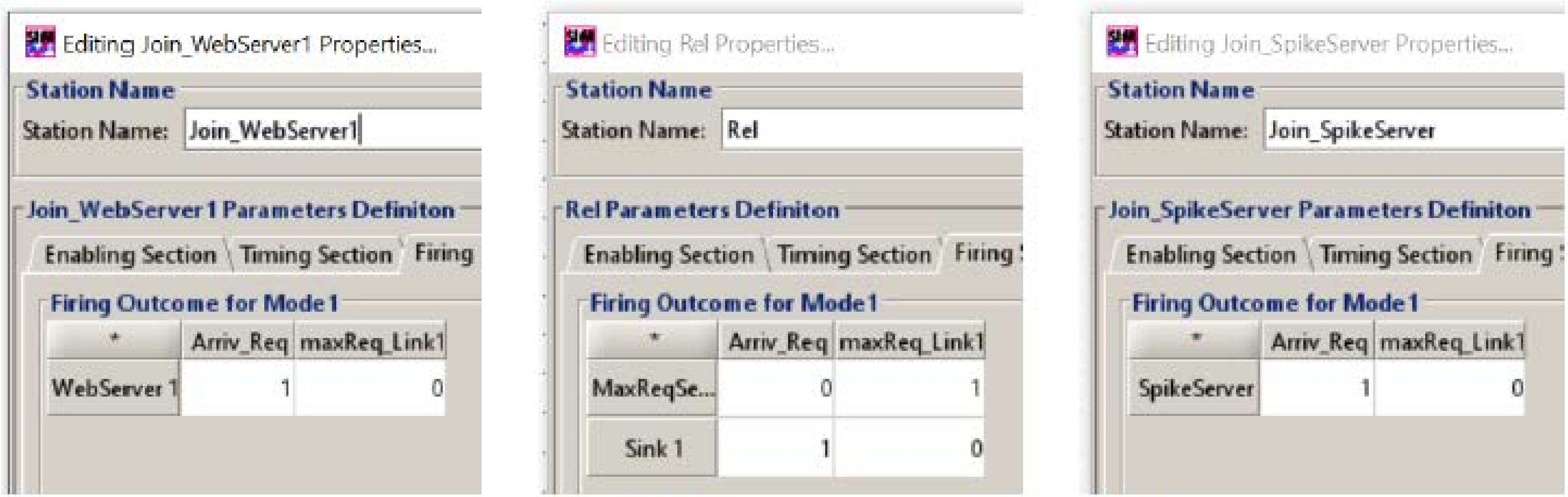
(b) Transition Rel



(c) Transition JoinSpikeserver

INHIBITING CONDITION: transition is blocked when there are 1 or more tokens in place MaxReqServer1

Firing rules of the three transitions

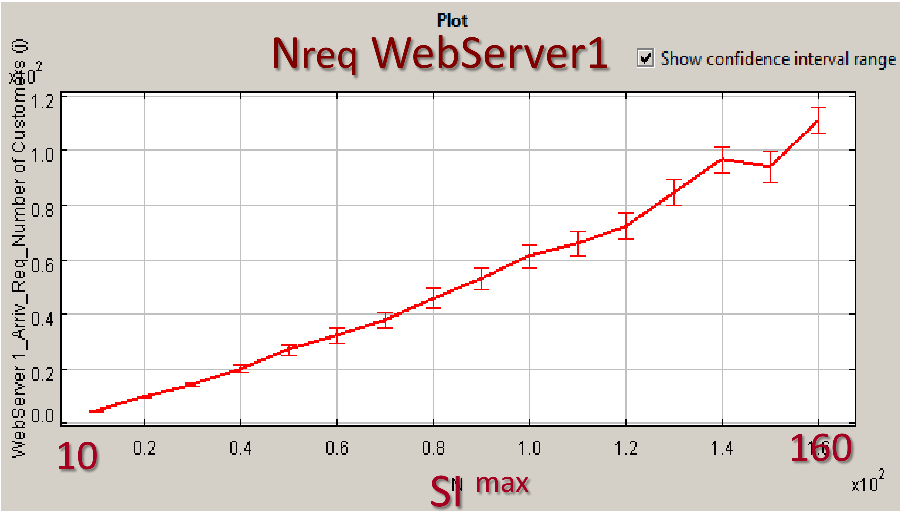


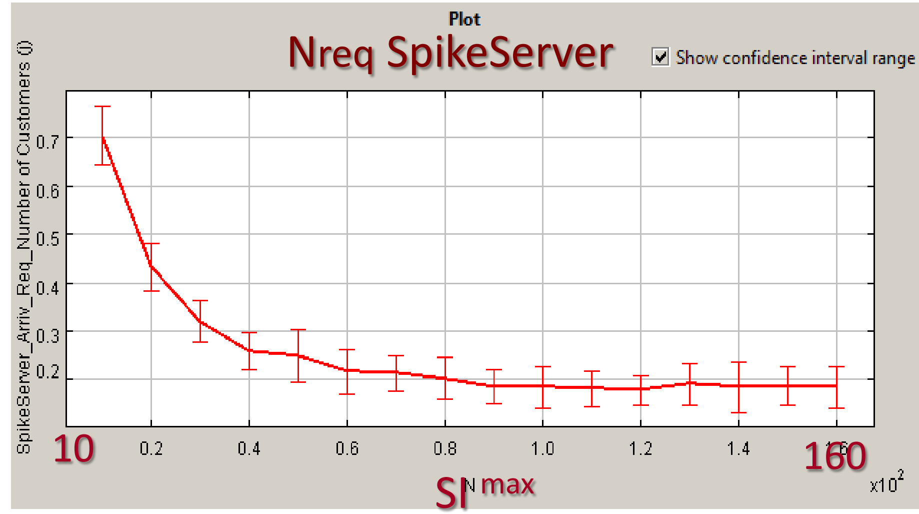
(a) Transition `Join WebServer1` (b) Transition `Rel` (c) Transition `JoinSpikeServer`

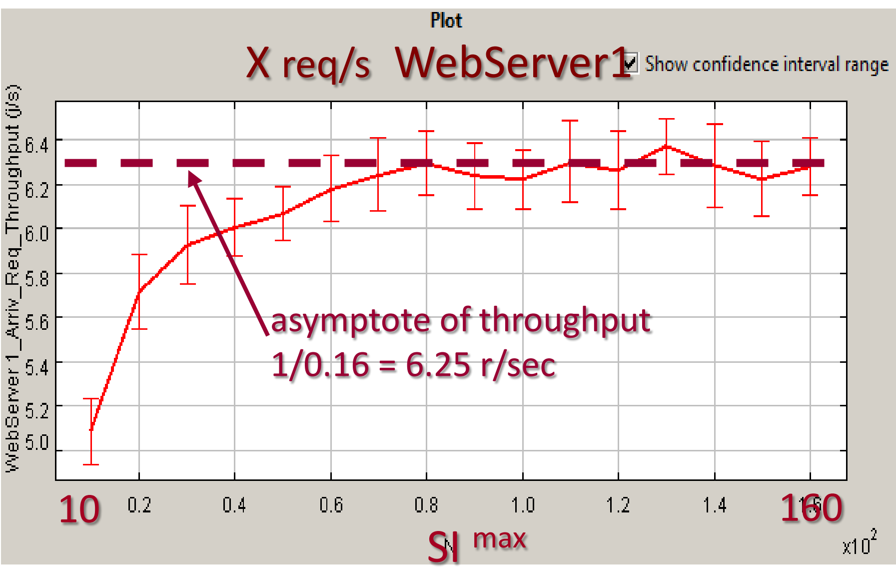
N requests and throughput *vs* alarm SI^{max} (**What-if**: SI^{max}=10÷160)

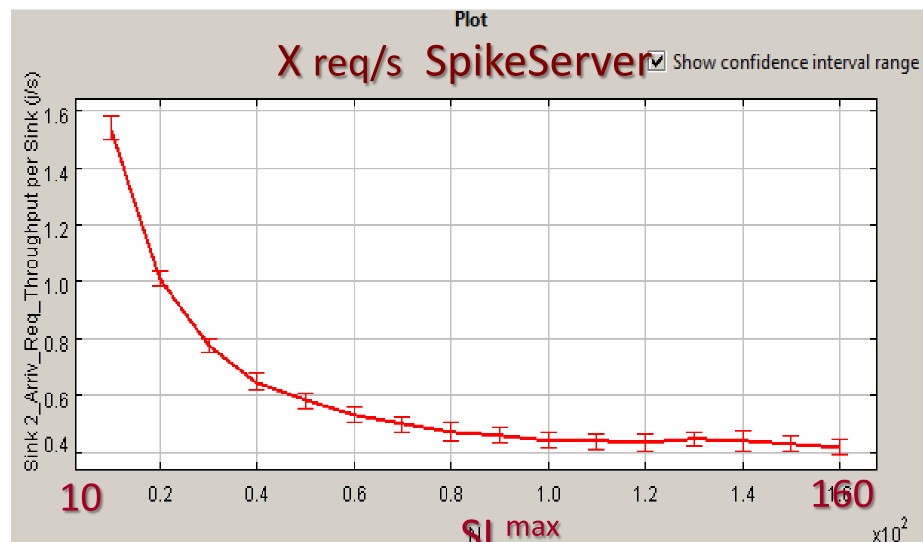
arrival rate=6.66 r/s hyper-exp cv=4

Service demand=160ms hyper-exp cv=4







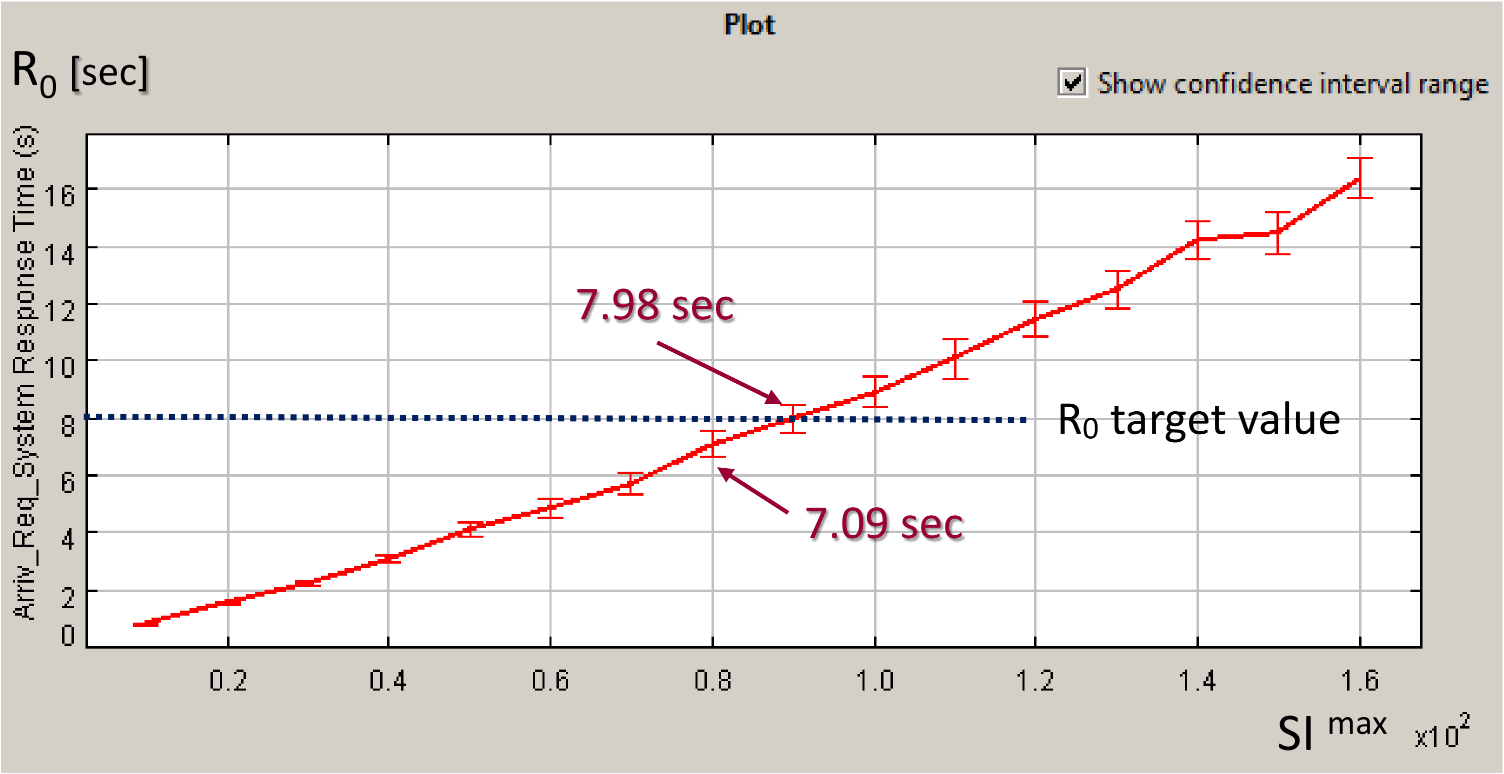


System response time R_0 *vs* SI^{max} (What-if: 10 ÷160 req)

fixed arrival rate = 6.66 req/sec

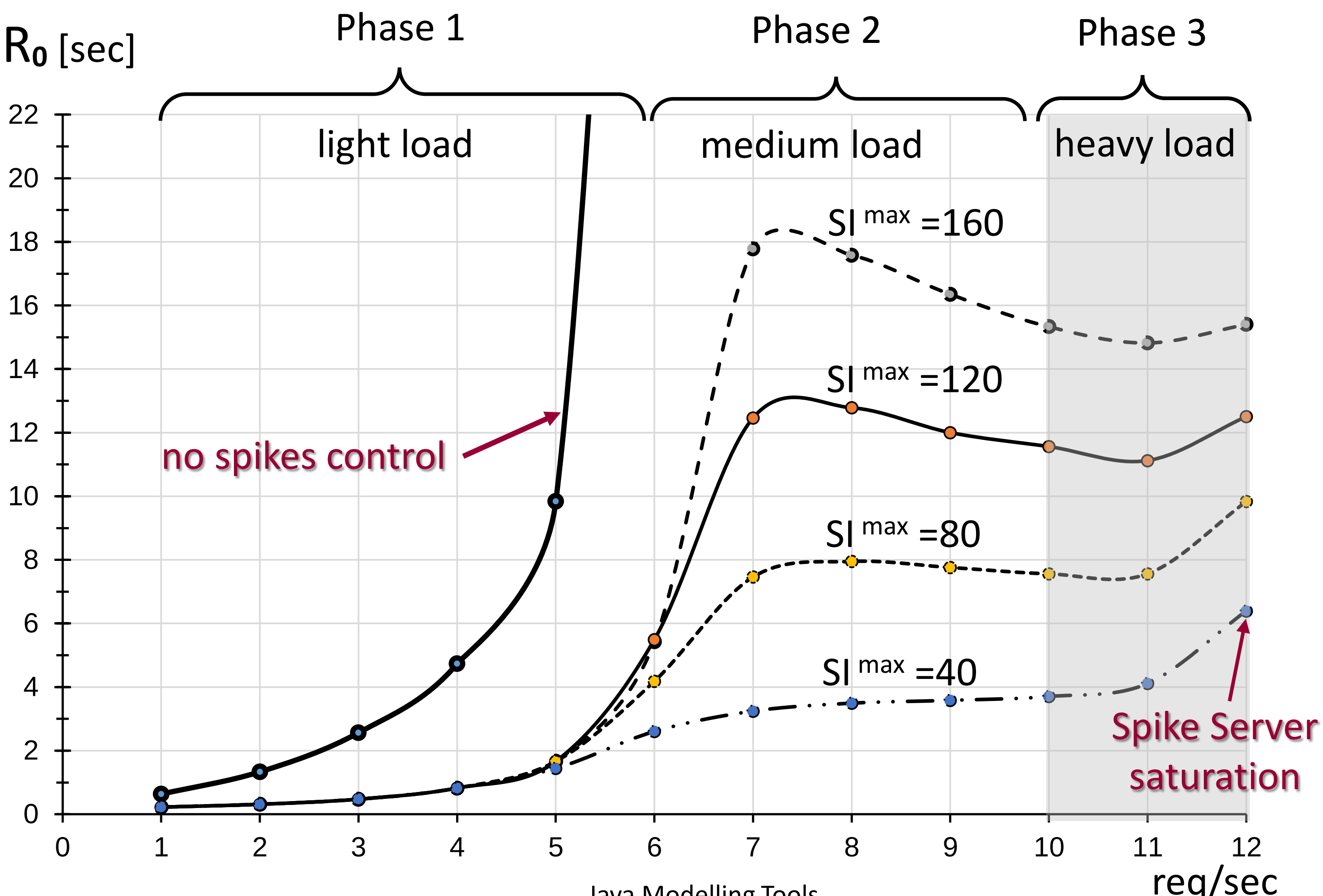
R_0 target value = 8 sec -> SI^{max}= 90 req R_0=7.98s , too close to the target

80 req is better --> R_0=7.09s)



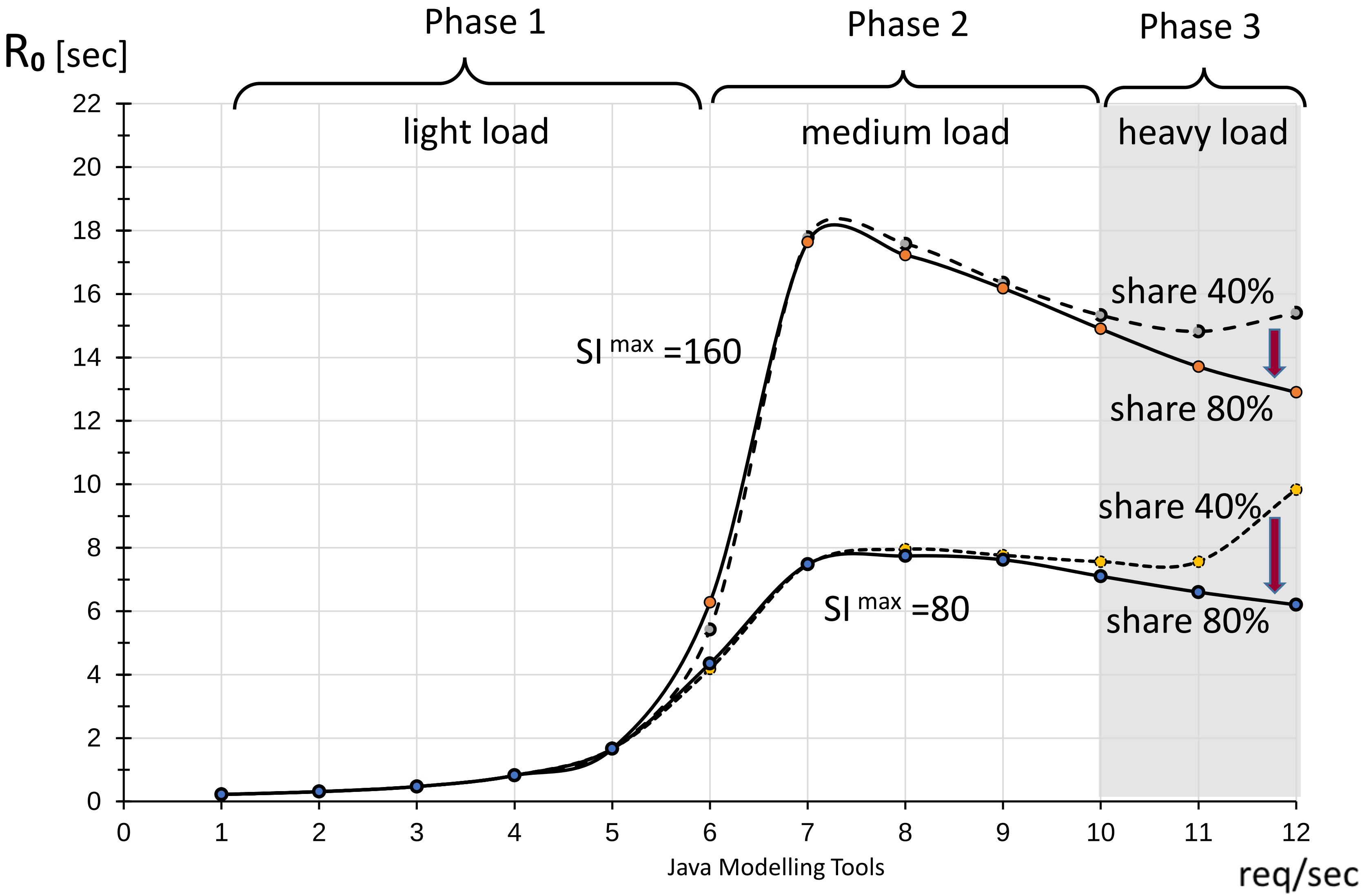
System response time R_0 *vs* arrival rate (SI^{max} 40÷160 req)

--- What-if: arr.rate = 1 ÷ 12 req/sec (60 ÷ 720 req/min) hyper-exp c=4
--- Service demands of both servers = 160 ms hyper-exp cv=4 (PS scheduling)
--- vertical asymptote of R_0 with no spikes control: 1/0.160 = 6.25 r/s



Vertical scaling of SpikeServer (VM with CPU share 40%->80%)

- service demands SpikeServer 160ms --> 80ms (share 80%)
- 12r/s SI^{max}=80: 40%share R_0=9.8s --> 80%share R_0=6.2s



System Response time R_0 *vs* arrival rate

- the trend of R_0 as the arrival rate increases shows three different operational phases with light, medium, heavy load

- Phase1 (1÷6 r/s): low arrival rates, the SpikeServer is practically not utilized, its contribution to the computation of R_0 is negligible
(6r/s, SI^{max}=80: 40%share U_{Spike}=0.01 R0=4.1s, 80%share U_{Spike}=0.007 R_0=4.3s)

- Phase2 (6÷10 r/s): the utilization of WebServer1 increases with the load routed to SpikeServer and thus its contribution to R_0
(10r/s, SI^{max}=80: 40%share U_{Spike}=0.59 R_0=7.5s, 80%share U_{Spike}=0.3 R_0=7.1s)

- Phase3 (10÷12 r/s): the response time of SpikeServer strongly influences the value of R_0 (note that its load are bursts!); this condition must be AVOIDED --> Vertical scaling SpikeServer or add a WebServer
(12r/s, SI^{max}=80: 40%share U_{Spike}=0.97 R_0=9.8s, 80%share U_{Spike}=0.45 R_0=6.2s)

positive effects of the SpikeServer

- smoothing effect on System response times: the high percentiles due to the high-load states of the WebServer are replaced by lower values obtained by the SpikeServer, which is typically not congested

- the mean System response time decreases, so will the number of scaling up actions as well

- as a consequence, the potential oscillations are also reduced

- by Vertical scaling the VM CPU share of SpikeServer it is possible to obtain a further decrease of Sys.resp.time R_0 for arrival rates 10÷12 req/sec (for higher rates the SpikeServer will saturate it too and then the Sys.resp.time will start increasing again)

- one SpikeServer can execute the spikes routed by several WebServer thus the costs for the execution of the global workload are reduced

Machine Learning applications

- most autoscaling problems can be solved with ML algorithms

- especially those of automatic tuning of the parameters of the autoscaling component:
  - identification of the max value of Spike Indicator SI that with a given workload satisfies the user's performance needs with the minimum number of allocated resources
  - identification of the size of the moving window for monitoring the performance indicators metrics
  - identification of the set of parameters that minimize oscillations
  - adaptivity to changes of workload characteristics and computational capacity of the servers
  - workload characterization (statistical parameters and fluctuations) and forecast for predictive scaling

Simulation of the Workflow of a Web App

Chapter 6 --- Sect.6.3

open model
three class workload
tool used: JSIMg (Queue Net with Class Switch)

Sect.6.3 - the problem

- we consider a simplified version of the e/commerce application of an online food shopping company
  - the web services of the app are allocated on two servers of private cloud
  - server A, a multicore system: front-end services, customer authentication, admin., CRM processes, interaction with payment service for strong authentication, check-out, update DB, invoice gen., shipping, tracking, ...
  - server B, a multiprocessor blade system, fault tolerant, with large RAM and SSD: browsing the catalog, processing shopping carts, manage of in-memory DB, ...
  - server P, external provider: payment services
- model the sequence of executions of services for an order submission
- evaluate the impact on performance of a new workload (15% higher than the current one and with a new service of Strong Customer Authentication for security payments)

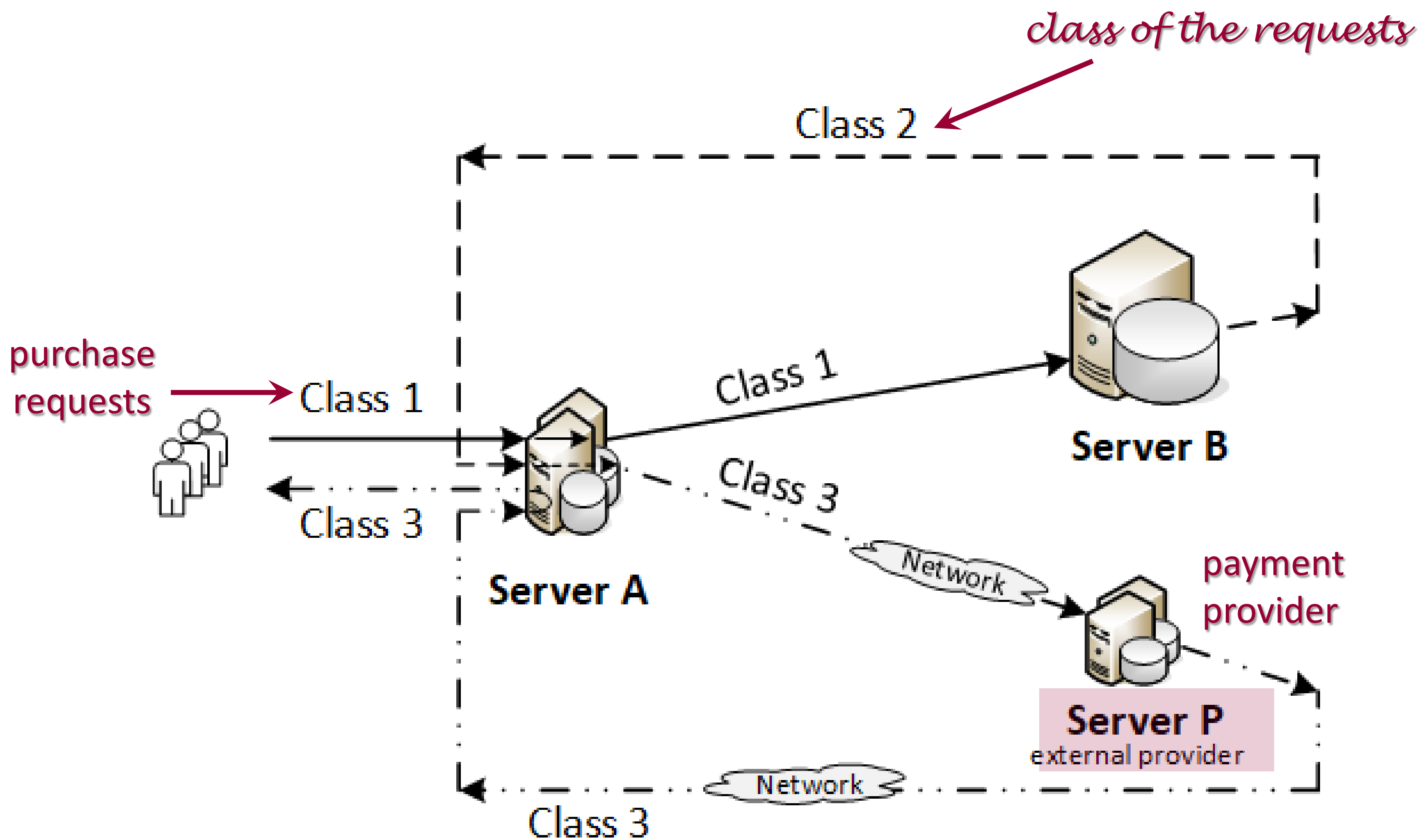
objectives

- model a given sequence of web service executions using the class-switch element
  - sequence of executions on the three servers A,B,P:

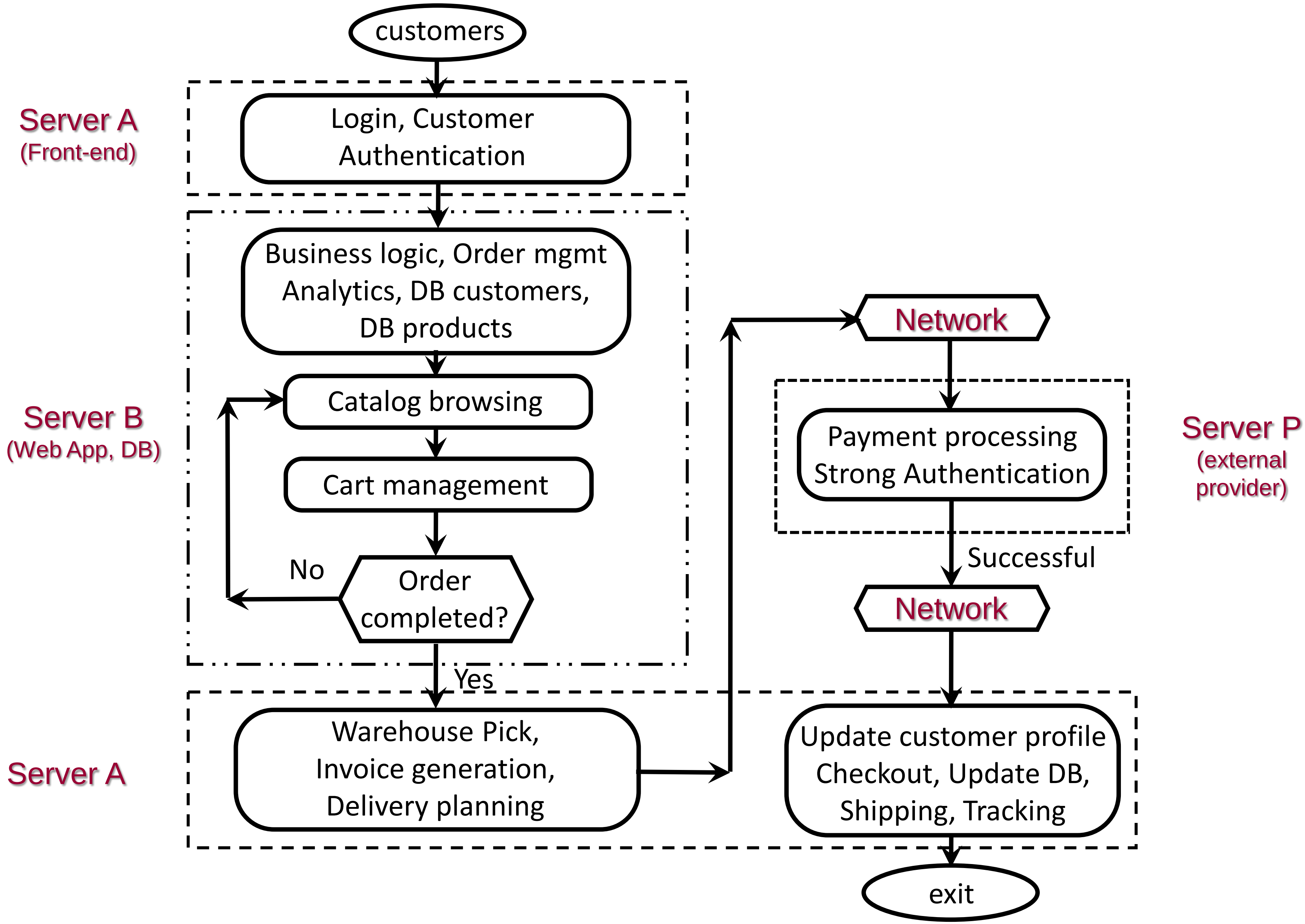
    sequence of visits A -> B -> A -> P -> A

- a request changes its class three times during execution
- evaluate system response time R and throughput X
  - with different arrival rates of requests
  - with two security payment algorithms for Strong Customer Authentication (SCA) (single factor, two-factor)
- new workload, arrival rate > 15% than current workload
  - increase throughput bound > 5000 req/hour (new Server B twice as fast)
  - impact on R and X
  - new bottleneck?

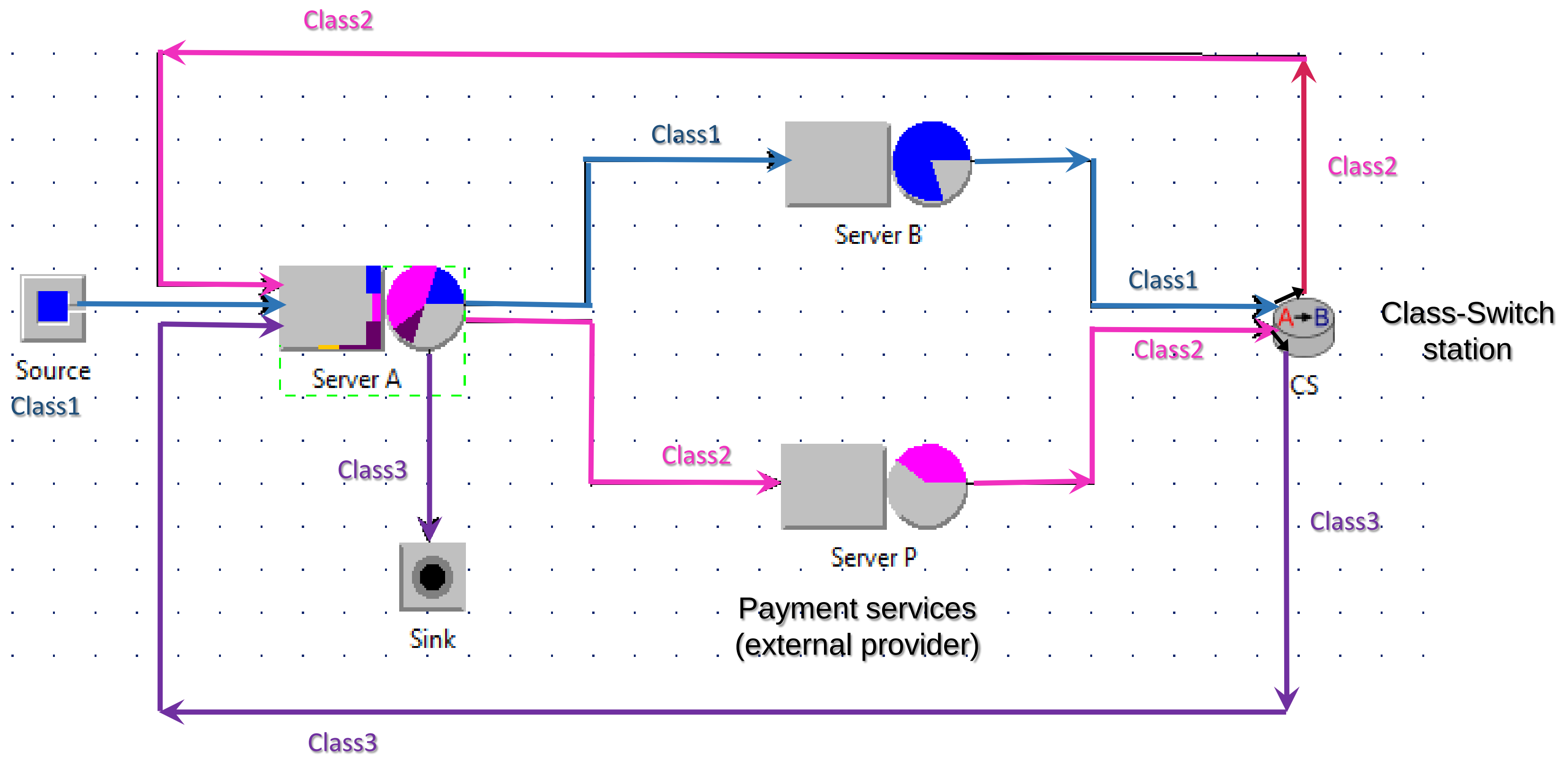
the data center



tasks of the workflow of the web app and servers used



the JSIMg model



Service demands D_i [sec]

Stations	Classes		
	1	2	3
Server A (Login, Front end, ...)	0.2	0.4	0.1
Server B (Web App Serv., DBs, ...)	0.8	0	0
Server P (Payment Provider)	0	0.4	0

(a)

with single-factor authentication

Stations	Classes		
	1	2	3
Server A (Login, Front end, ...)	0.2	0.4	0.15
Server B (Web App Serv., DBs, ...)	0.8	0	0
Server P (Payment Provider)	0	0.7	0

(b)

with two-factor authentication for secure payment

Class Switch probabilities

CS Parameters Definiton

Class Switch Matrix | Routing Section

CS Strategies

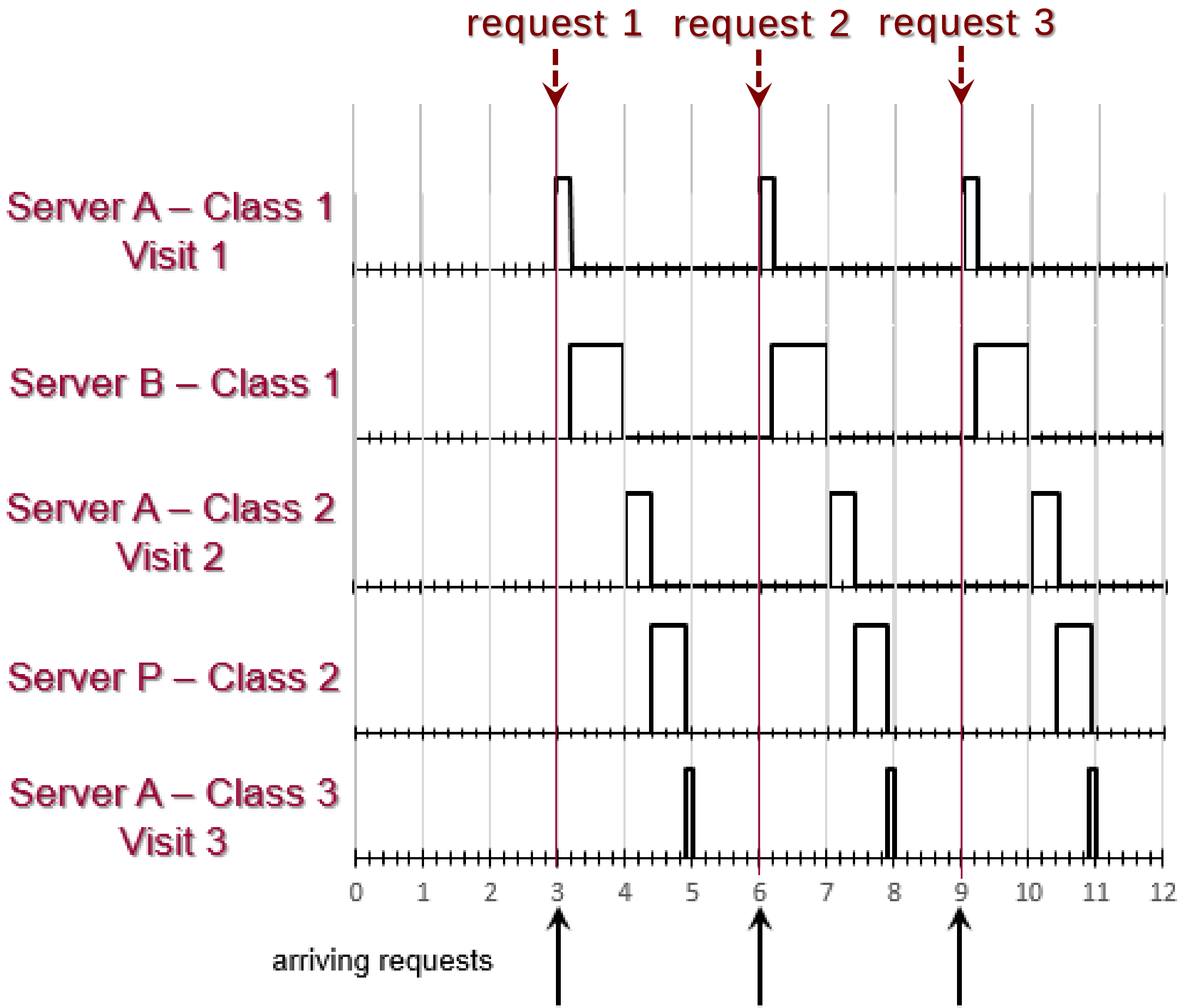
*	Class1	Class2	Class3
Class1	0.0 (0%)	1.0 (100%)	0.0 (0%)
Class2	0.0 (0%)	0.0 (0%)	1.0 (100%)
Class3	0.0 (0%)	0.0 (0%)	1.0 (100%)

class of incoming requests →

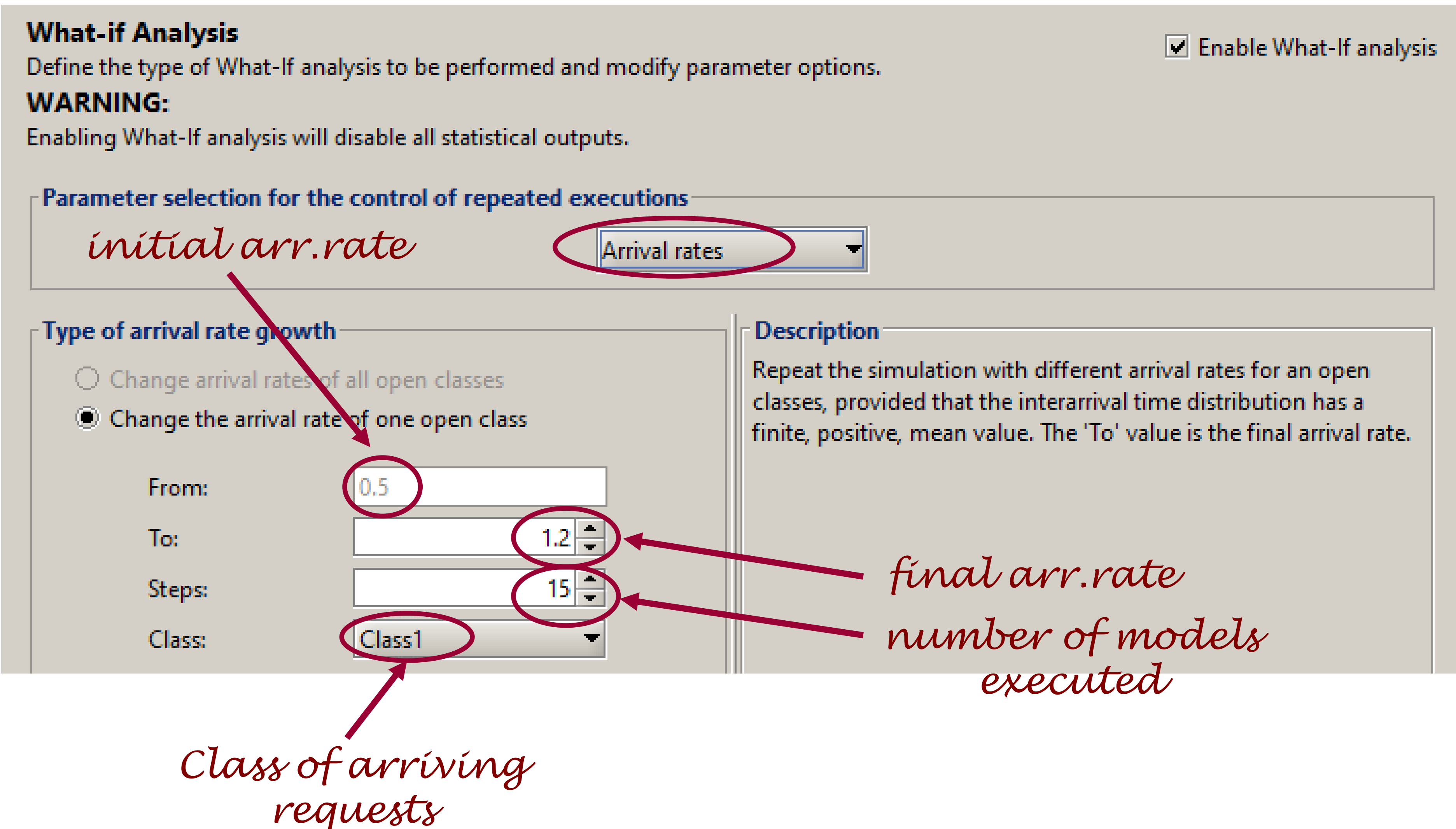
→ class of outgoing requests

probabilities of class switch

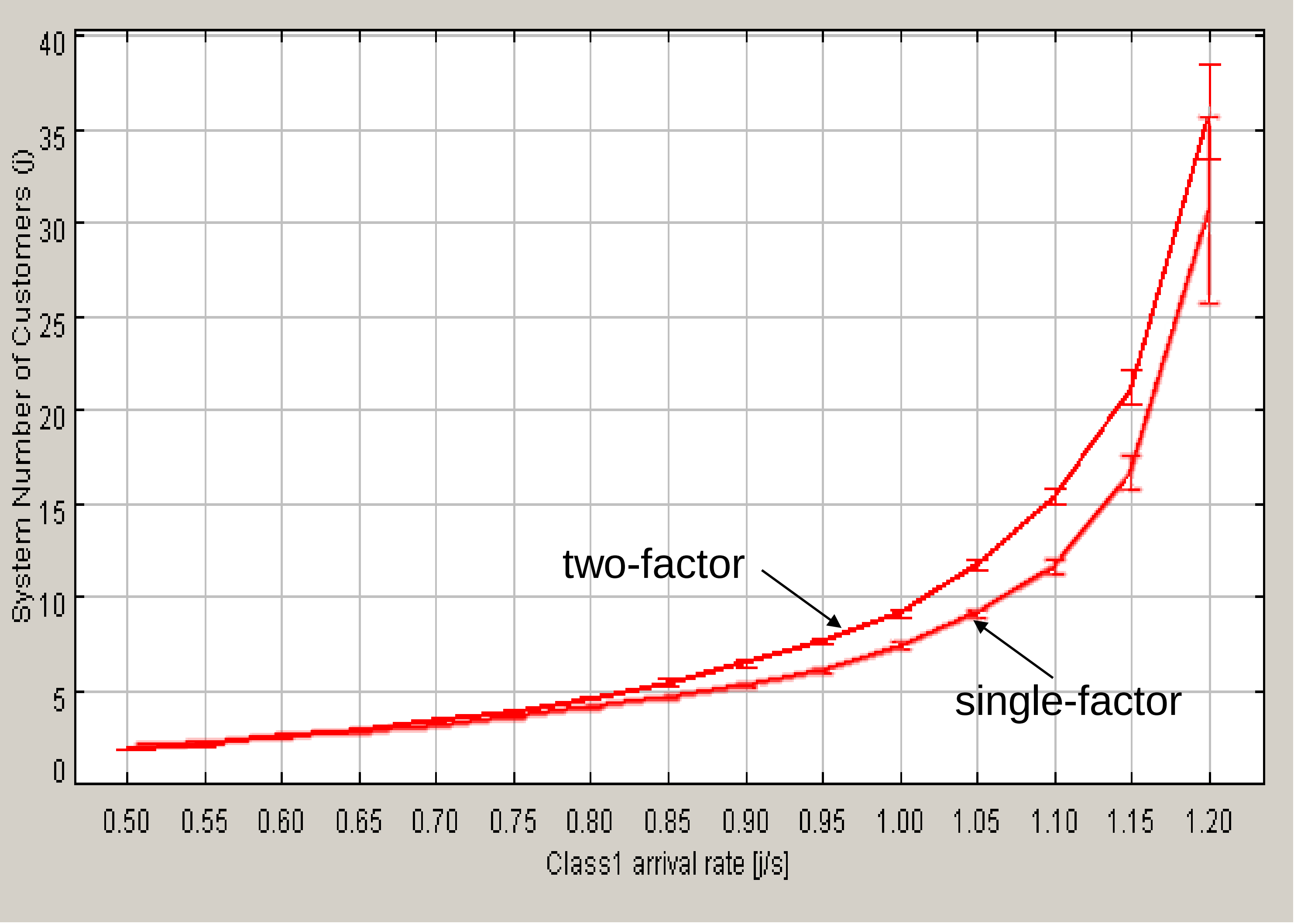
temporal sequence of the execution of three requests
(deterministic times)



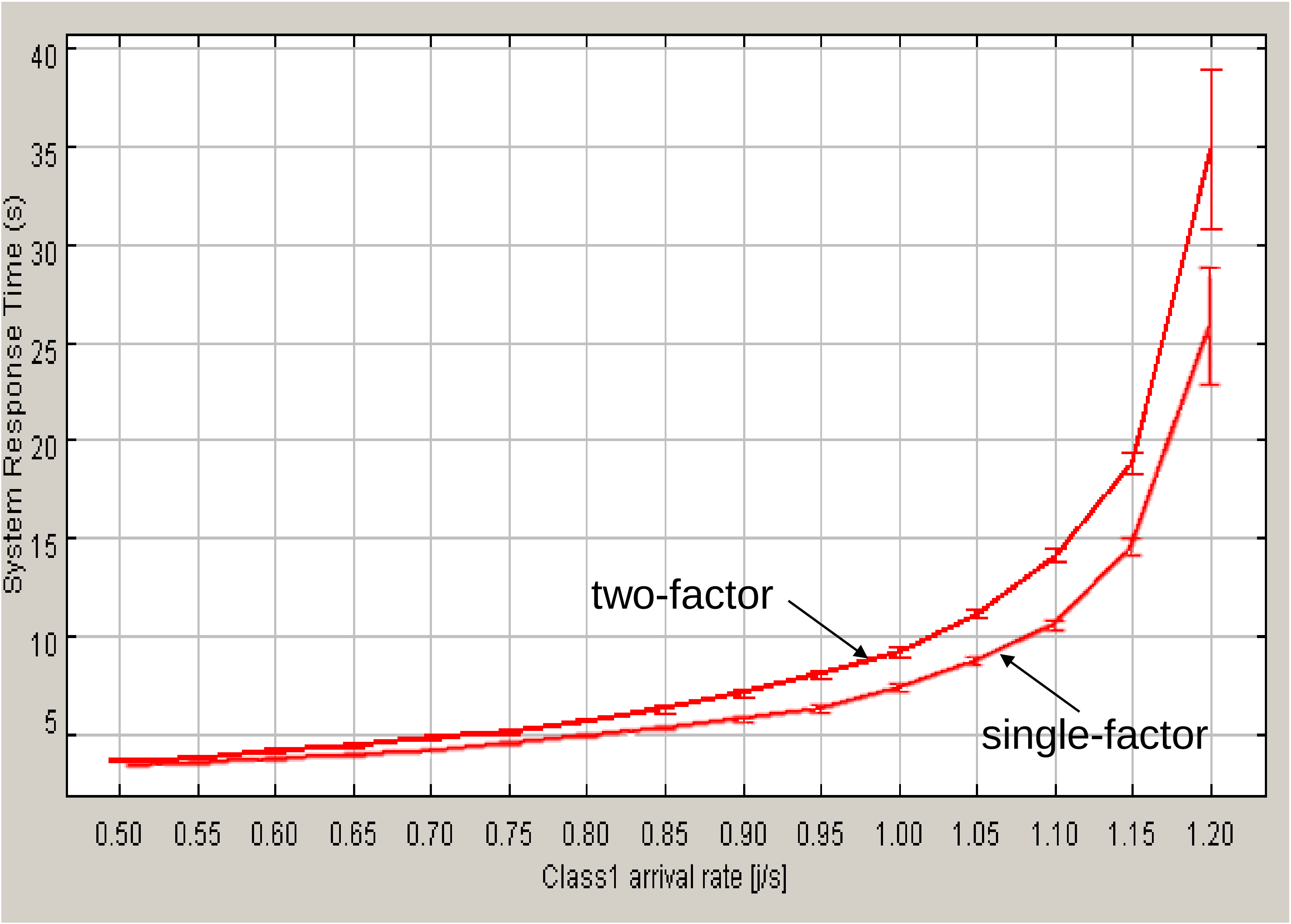
What-if parameters (λ=0.5÷1.2 req/sec)



N with single-factor and two-factor authentications



R with single-factor and two-factor authentications



R with OLD and NEW Server B (new throughput bound)