Institute of Computer Science
Chair of Communication Networks
Prof. Dr. Tobias Hoßfeld

Discrete-Event Simulation using R-Simmer

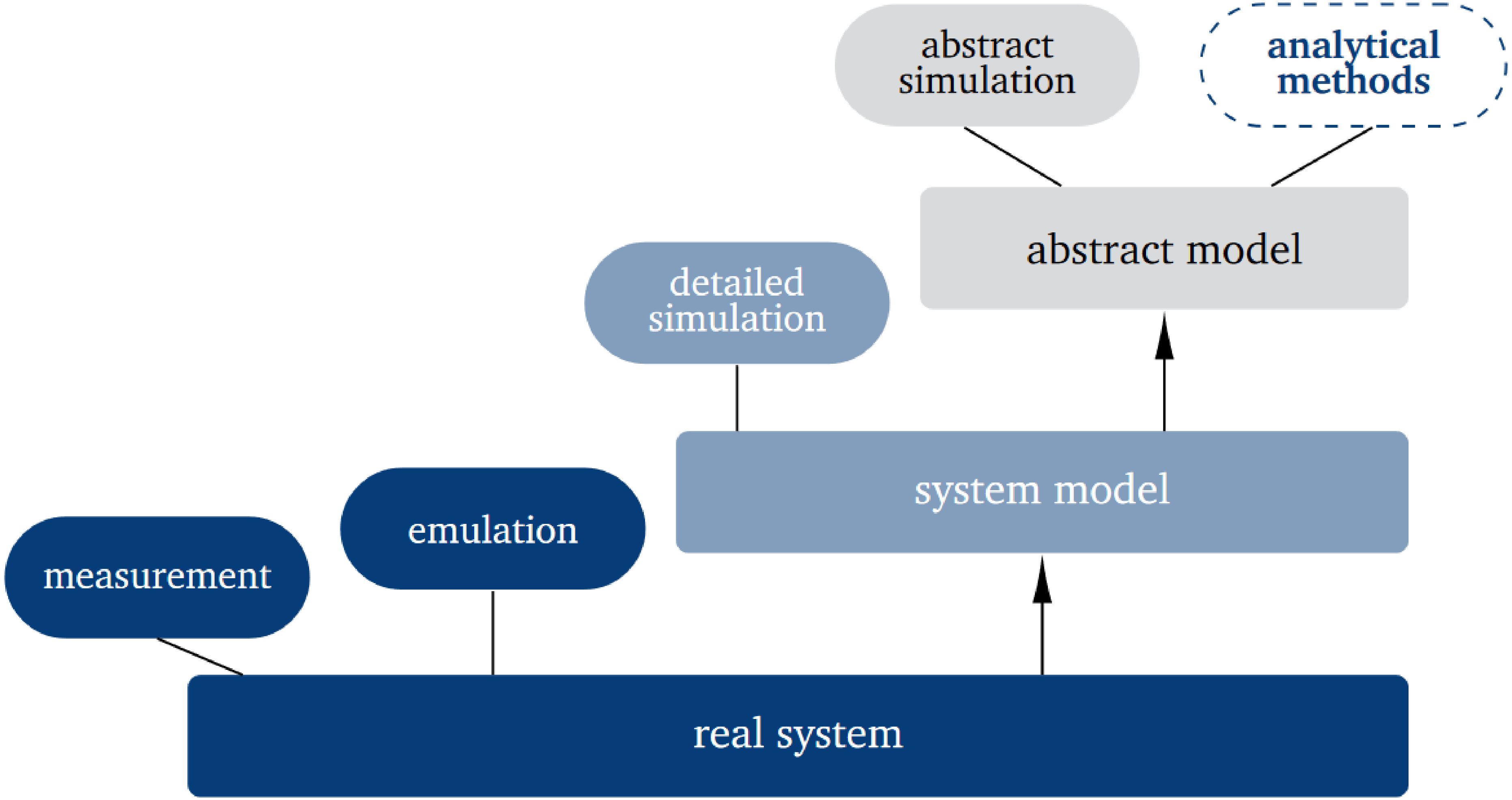
KuVS Fachgespräch – WueWoWAS'22 – Würzburg

Stefan Geißler
stefan.geissler@uni-wuerzburg.de

From R. Shannon (1975), simulation is

the process of designing a ***model of a real system*** *and conducting experiments with this model for the purpose either of* ***understanding the behavior of the system*** *or of* ***evaluating various strategies*** *[...] for the operation of the system.*

Shannon, Robert E. Systems simulation; the art and science. No. 04; T57. 62, S4.. 1975.

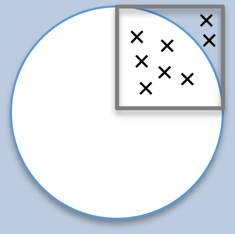


Tran-Gia, Phuoc, and Tobias Hoßfeld. Performance Modeling and Analysis of Communication Networks: A Lecture Note. BoD–Books on Demand, 2021.

Classification of Simulation Models

Static

- Simulation of a system at exactly one point in time (e.g. Monte Carlo Simulation)



Deterministic

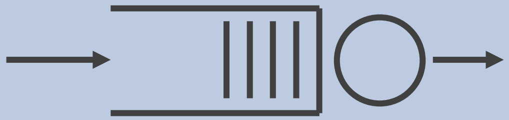
- Simulation of a system without randomization (e.g. differential equations for chemical processes)

Continuous

- Simulation of a system that exhibits state changes at continuous points in time (e.g. chemical reactions)

Dynamic

- Simulation of a systems behavior over time (e.g. queueing models)



Stochastic

- Simulation of a system under the influence of stochastic processes (e.g. queueing models)

Discrete

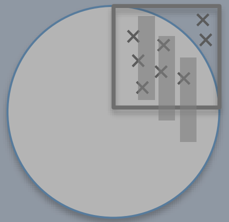
- Simulation of a system that exhibits state changes at discrete points in time (e.g. queueing models)

Law, Averill M., W. David Kelton, and W. David Kelton. Simulation modeling and analysis. Vol. 3. New York: Mcgraw-hill, 2007.

Classification of Simulation Models

Static

- Simulation of a system at exactly one point in time (e.g. Monte Carlo Simulation)



Deterministic

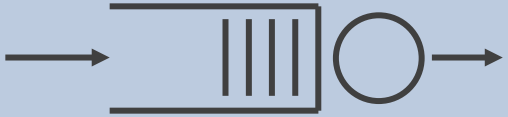
- Simulation of a system without randomization (e.g. differential equations for chemical processes)

Continuous

- Simulation of a system that exhibits state changes at continuous points in time (e.g. chemical reactions)

Dynamic

- Simulation of a systems behavior over time (e.g. queueing models)



Stochastic

- Simulation of a system under the influence of stochastic processes (e.g. queueing models)

Discrete

- Simulation of a system that exhibits state changes at discrete points in time (e.g. queueing models)

Law, Averill M., W. David Kelton, and W. David Kelton. Simulation modeling and analysis. Vol. 3. New York: Mcgraw-hill, 2007.

Classification of Simulation Models

Static

▶ Simulat
one po
Simulat

Deterministic

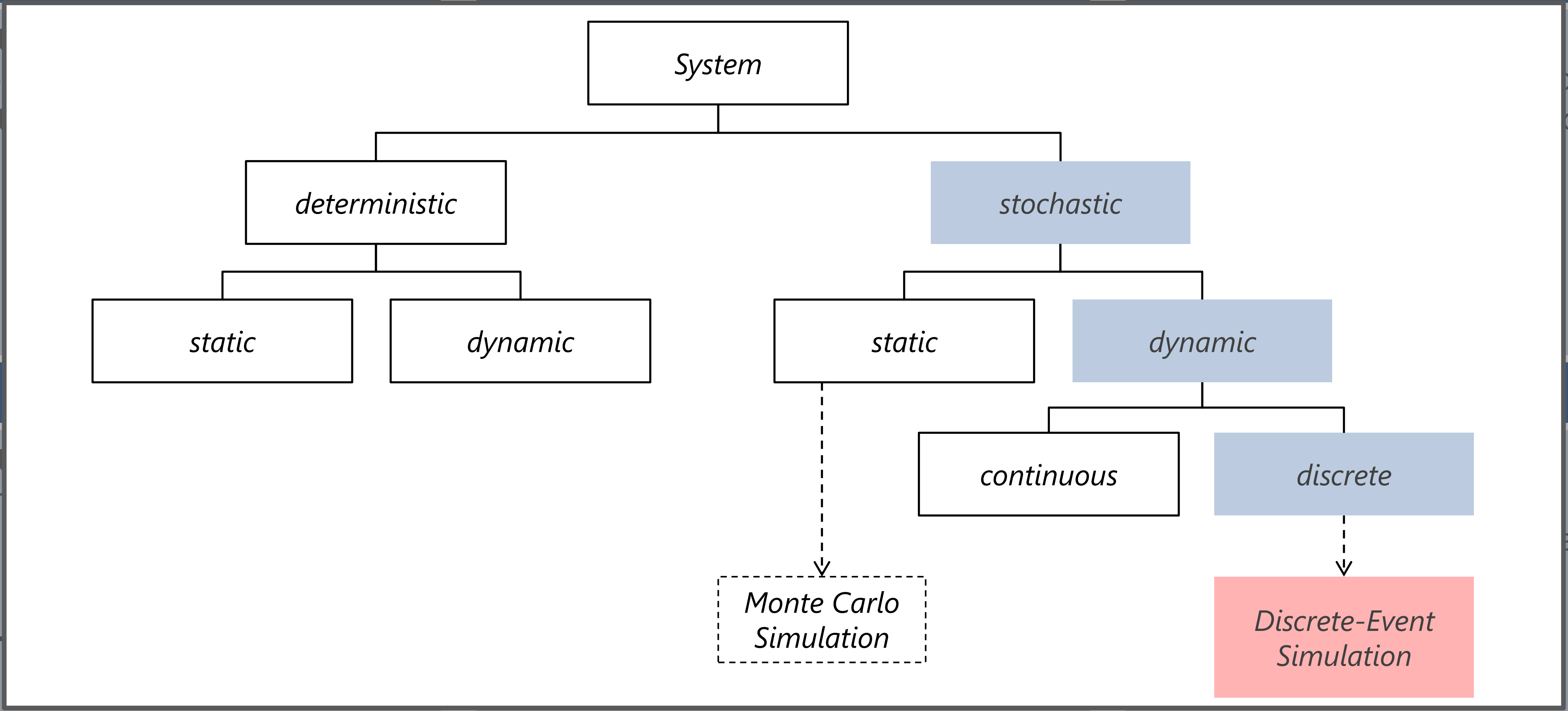
Continuous

at exhibits
ous points
ctions)

Dynamic

▶ Simulat
over ti

at exhibits
points in
els)



Law, Averill M., W. David Kelton, and W. David Kelton. Simulation modeling and analysis. Vol. 3. New York: Mcgraw-hill, 2007.

Abstract Idea of a Simulation

- Base for most simulations is a real world system – e.g. Application, Hardware Component, Distributed System, Industry Process

Trigger ⟷ **Resource** ⟷ **Result**

Random or Periodic Event
- Packet Arrival
- System Signal
- User Interaction

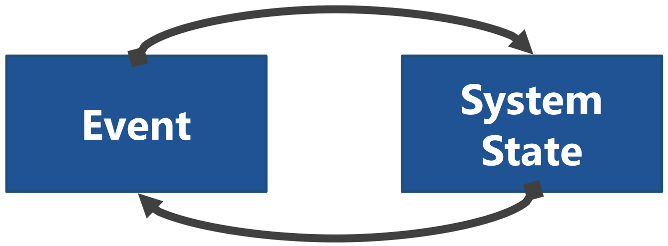
Physical or Logical Component of Real World System
- CPU, Memory
- Network Link
- Human Resource

Near Arbitrary System Reaction Depending on Trigger Type
- Processing End
- Packet Drop
- CPU Interrupt
- Link Congestion
- Event Mishandling

- Representation of real world system as abstracted sequence of events

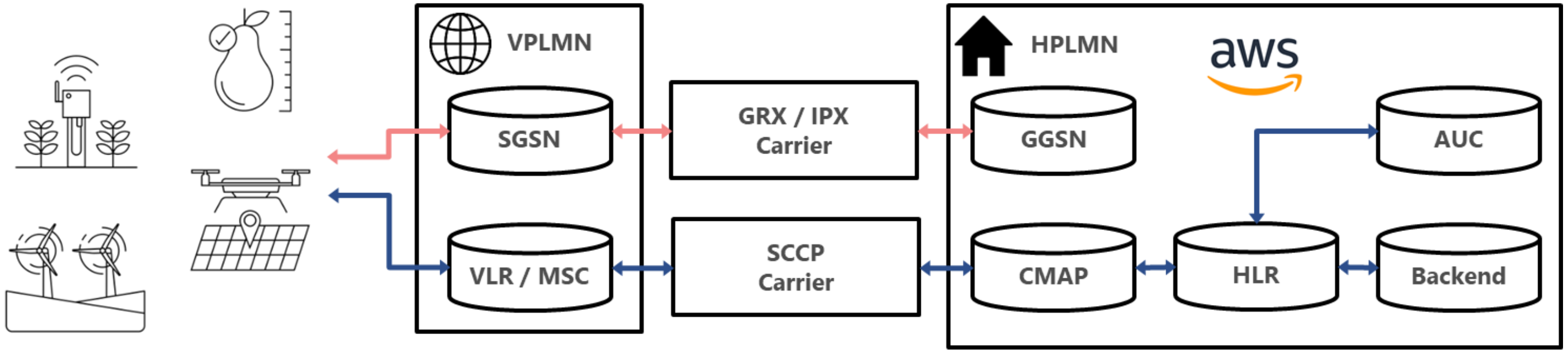
➔ **(Almost) anything can be simulated!**

➔ **How close to reality can we get?**



Simple Queueing System

Complex Mobile Communication System



simmer: Discrete-Event Simulation for R

Iñaki Ucar
Universidad Carlos III de Madrid

Bart Smeets
dataroots

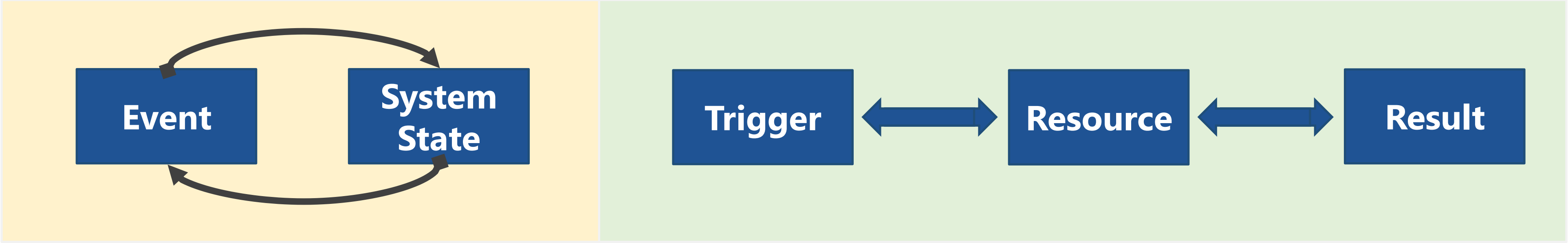
Arturo Azcorra
Universidad Carlos III de Madrid
IMDEA Networks Institute

Abstract

The **simmer** package brings discrete-event simulation to R. It is designed as a generic yet powerful process-oriented framework. The architecture encloses a robust and fast simulation core written in C++ with automatic monitoring capabilities. It provides a rich and flexible R API that revolves around the concept of *trajectory*, a common path in the simulation model for entities of the same type.

Resources
https://r-simmer.org
https://www.rdocumentation.org/packages/simmer/versions/4.4.2
https://groups.google.com/g/simmer-devel
https://github.com/r-simmer/simmer

Advanced concepts (e.g. replication, parallelization):
https://r-simmer.org/slides/20191115_xijur_simmer



- **Events** can be triggered at any time
- **Resources** have limited **capacity** and may have a limited **queue size**

```r
library(simmer)

paket.trajectory <- trajectory() %>%
  seize("resource") %>%
  timeout(5) %>%
  release("resource")

env <- simmer() %>%
  add_generator(name = "event", trajectory = paket.trajectory, distribution = at(2,4,6)) %>%
  add_resource(name = "resource", capacity = 1, queue_size = Inf, mon = T)

run(env)

```

- **Generators** produce new events
- **Resources** process events based on their capacity
- **Trajectories** define interactions between events and resources as well as other events

The simmer API - Trajectories

- **trajectories** are recipes, or lists, of activities that define the life time of arrivals
- Similar to **dplyr** for data manipulation

From H. Wickham

[…] by constraining your options, it simplifies how you can think about [something]

- Fixed parameters

```
traj0 <- trajectory() %>%
  log_("Entering the trajectory") %>%
  timeout(10) %>%
  log_("Leaving the trajectory")
```

- Dynamic parameters

```
traj1 <- trajectory() %>%
  log_(function() "Entering the trajectory") %>%
  timeout(function() 10) %>%
  log_(function() "Leaving the trajectory")
```

```r

paket.trajectory <- trajectory() %>%
  seize("resource") %>%
  timeout(5) %>%
  release("resource")

```

Full reference can be found online:
https://r-simmer.org/reference/

- **trajectory()** defines a sequence of actions triggered by an event
  - Is generally processed step by step
  - Can be **rolled back** to create loops
  - Can **branch** or be **cloned**

- Basic **actions** to use in a trajectory are
  - **seize()** requests capacity from a resource
  - **timeout()** lets time pass
  - **release()** free previously requested capacity
  - **log_()** prints a log message
  - **set_attribute(), get_attribute()** sets/gets an arrival level attribute
  - **set_global(), get_global()** sets/gets a global attribute
  - **rollback()** rolls back the current trajectory by a specific amount of actions

The simmer API – Generators and Resources

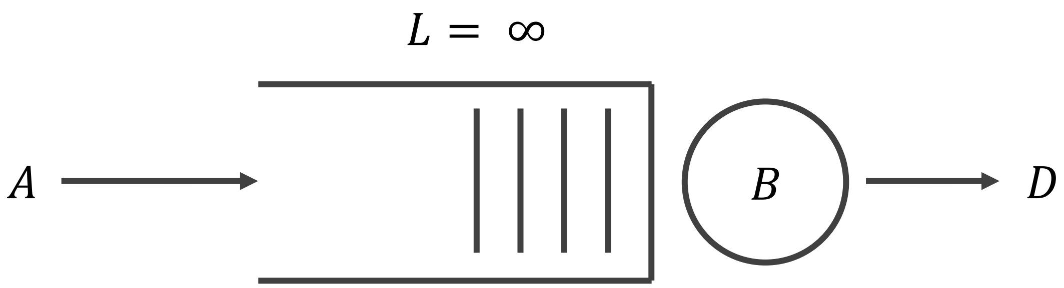
```r

env <- simmer() %>%
  add_generator(name = "event", trajectory = paket.trajectory, distribution = at(2,4,6)) %>%
  add_resource(name = "resource", capacity = 1, queue_size = Inf, mon = T)

```

- **add_generator()** defines a source of events or arrivals
  - **name** is the canonical name of the resource
  - **trajectory** defines what generated events do
  - **distribution** dictates when new events occur

- **add_resource()** defines a claimable resource
  - **name** is the canonical name of the resource
  - **capacity** is the number of available processing
  - **queue_size** is the maximum number of elements allowed to wait before rejects occur
  - **mon** defines whether the resource needs to be monitored

The Gi/Gi/1-∞ Queue

$$a(4) = 0.4, \quad a(8) = 0.5, \quad a(20) = 0.1, \quad a(k) = 0 \text{ otherwise},$$
$$b(4) = 0.2, \quad b(5) = 0.1, \quad b(6) = 0.7, \quad b(k) = 0 \text{ otherwise}.$$
$$\mathrm{E}[A] = 7.6, \quad \mathrm{E}[B] = 5.5, \quad \rho = 0.724, \quad c_A = 0.598, \quad c_B = 0.147.$$



*Hands-On: ex1/**ex1.R***
- *Define simple trajectory*
- *Define generators*
- *Define resources*

Can we obtain similar results as Tobias?

- **add_generator()** defines a source of events or arrivals
  - **name** is the canonical name of the resource
  - **trajectory** defines what generated events do
  - **distribution** dictates when new events occur

- **add_resource()** defines a claimable resource
  - **name** is the canonical name of the resource
  - **capacity** is the number of available processing
  - **queue_size** is the maximum number of elements allowed to wait before rejects occur
  - **mon** defines whether the resource needs to be monitored

- **trajectory()** defines a sequence of actions triggered by an event
  - Is generally processed step by step
  - Can be **rolled back** to create loops
  - Can **branch** or be **cloned**

- Basic **actions** to use in a trajectory are
  - **seize()** requests capacity from a resource
  - **timeout()** lets time pass
  - **release()** free previously requested capacity
  - **log_()** prints a log message
  - **set_attribute(), get_attribute()** sets/gets an arrival level attribute
  - **set_global(), get_global()** sets/gets a global attribute
  - **rollback()** rolls back the current trajectory by a specific amount of actions

simmer
DES for R

*Hands-On: ex1/**ex1.R***
- *Define simple trajectory*
- *Define generators*
- *Define resources*

- **add_generator()** defines a source of events or arrivals
  - **name** is the canonical name of the resource
  - **trajectory** defines what generated events do
  - **distribution** dictates when new events occur

- **add_resource()** defines a claimable resource
  - **name** is the canonical name of the resource
  - **capacity** is the number of available processing
  - **queue_size** is the maximum number of elements allowed to wait
  - **mon** defines wh… monitored

- **trajectory()** defines a sequence of actions triggered by an event
  - Is generally processed step by step
  - Can be **rolled back** to create loops
  - Can **branch** or be **cloned**

- Basic **actions** to use in a trajectory are
  - **seize()** requests capacity from a resource
  - **timeout()** lets time pass
  - …ted capacity
  - … sets/gets an arrival
  - …gets a global attribute
  - … trajectory by a specific amount of actions

How to adapt the simulation to be a Gi/Gi/1-L with L = 10 ?
How to adapt the simulation to be an M/M/1-L model ?
What happens if E[A] < E[B] ?

*Hands-On: ex1/**ex1.R***
- *Define simple trajectory*
- *Define generators*
- *Define resources*

*Hands-On: **video_streaming/001_basic.R***

- *Basic structure of a more complex simulation*
- *Video streaming application with limited bandwidth*
- *Clients request video based on randomized video bitrate*

*Hands-On: **video_streaming/002_infrastructure.R***

- *Extension of 001_basic.R*
- *More detailed application infrastructure*
- *Frontend, backend and streaming components simulated separately*

*Hands-On: **video_streaming/003_clients.R***

- *Extension of 002_infrastructure.R*
- *More detailed client behavior*
- *Clients browse, renege and exhibit rudimentary DASH behavior*