Introduction to Discrete-Event Simulation Using SimPy

Chun-Chieh Huang

DSP and Algorithm Design Department
Metanoia Communications Inc.

October 17, 2011

Outline

What is Simulation and Why do we need it?

- Simulation
  - a computer program that creates a virtual environment in order to study physical problems

What is Simulation and Why do we need it?

- Simulation
  - a computer program that creates a virtual environment in order to study physical problems
- When to use simulation
  - hard to do real experiment,
    e.g. battle field, or banking system

What is Simulation and Why do we need it?

- Simulation
  - a computer program that creates a virtual environment in order to study physical problems
- When to use simulation
  - hard to do real experiment,
    e.g. battle field, or banking system
  - cheaper to do simulation,
    e.g. RTL simulation for IC design,
    or highway/freeway route planning

What is Simulation and Why do we need it?

- Simulation
  - a computer program that creates a virtual environment in order to study physical problems
- When to use simulation
  - hard to do real experiment,
    e.g. battle field, or banking system
  - cheaper to do simulation,
    e.g. RTL simulation for IC design,
    or highway/freeway route planning
  - analyzing bottleneck for current workflow

What is Simulation and Why do we need it?

- Simulation
  - a computer program that creates a virtual environment in order to study physical problems
- When to use simulation
  - hard to do real experiment,
    e.g. battle field, or banking system
  - cheaper to do simulation,
    e.g. RTL simulation for IC design,
    or highway/freeway route planning
  - analyzing bottleneck for current workflow
- When not to use simulation
  - more expensive to do simulation,
    e.g. simple harmonic motion

What is Simulation and Why do we need it?

- Simulation
  - a computer program that creates a virtual environment in order to study physical problems
- When to use simulation
  - hard to do real experiment,
    e.g. battle field, or banking system
  - cheaper to do simulation,
    e.g. RTL simulation for IC design,
    or highway/freeway route planning
  - analyzing bottleneck for current workflow
- When not to use simulation
  - more expensive to do simulation,
    e.g. simple harmonic motion
  - problems that can be analyzed by pencil and paper

Categories of Simulation

- Continuous or discrete
  - State variable is continuous, e.g. weather systems
  - State variable is discrete, e.g. number of customers

Categories of Simulation

- Continuous or discrete
  - State variable is continuous, e.g. weather systems
  - State variable is discrete, e.g. number of customers
- Static or dynamic
  - Static: represents a system at a particular point of time
    - called Monte-Carlo Simulation [2]
  - Dynamic: represents systems as they change over time
    - e.g. banking system from 9:00 AM to 5:00 PM

Categories of Simulation

- Continuous or discrete
  - State variable is continuous, e.g. weather systems
  - State variable is discrete, e.g. number of customers
- Static or dynamic
  - Static: represents a system at a particular point of time
    - called Monte-Carlo Simulation [2]
  - Dynamic: represents systems as they change over time
    - e.g. banking system from 9:00 AM to 5:00 PM
- Deterministic or stochastic
  - Deterministic: contains no random variable
  - Stochastic: has one or more random variables

What is Discrete-Event Simulation?

- Simulation of weather system is **continuous**.
- Simulation of queue in a post office is **discrete**.
  - Number of customers in any time is discrete.
  - Simulation for this kind of systems is called discrete-event simulation.
- Mostly, but not limited to, queueing systems
  - factory work flow
  - freeway traffic simulation
  - network traffic simulation

Discrete-Event Simulation World Views

- Activity-oriented
  - fixed increment of time
  - time-consuming

Discrete-Event Simulation World Views

- Activity-oriented
  - fixed increment of time
  - time-consuming
- Event-oriented
  - on each event, generate next event and put into event queue
  - simulation time advances to next event
  - faster than activity-oriented

Discrete-Event Simulation World Views

- Activity-oriented
  - fixed increment of time
  - time-consuming
- Event-oriented
  - on each event, generate next event and put into event queue
  - simulation time advances to next event
  - faster than activity-oriented
- Process-oriented
  - abstract one object into a process
  - easier to maintain in the end

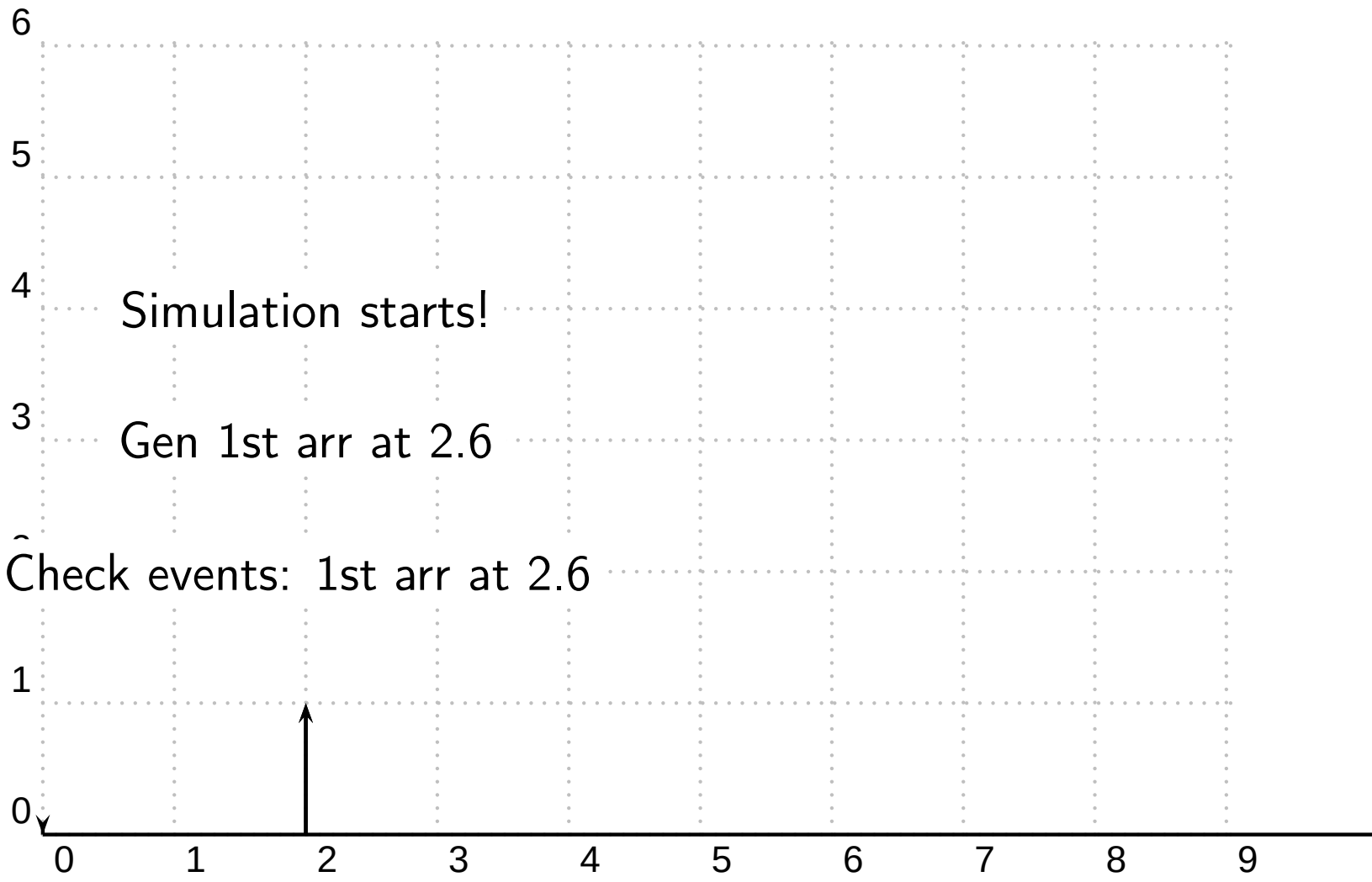
Example to Illustrate World Views

- Simulating a post office with only one clerk
- Customers come in at random time
  and wait if the clerk is already serving
- Clerk serves each customer for a random period of time

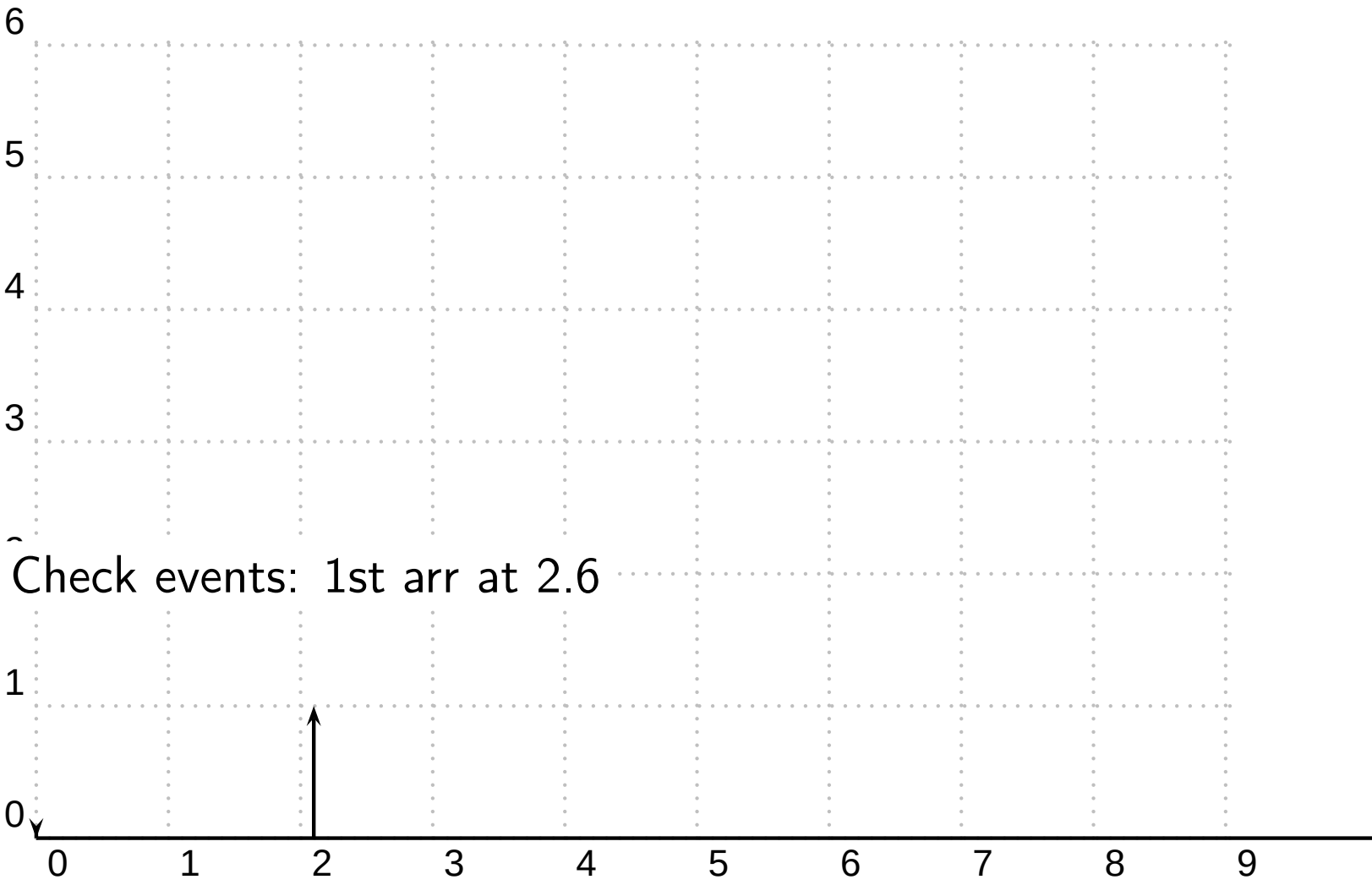
Discrete-Event Simulation World Views

- Activity-oriented
  - fixed increment of time
  - time-consuming
- Event-oriented
  - on each event, generate next event and put into event queue
  - simulation time advances to next event
  - faster than activity-oriented
- Process-oriented
  - abstract one object into a process
  - easier to maintain in the end

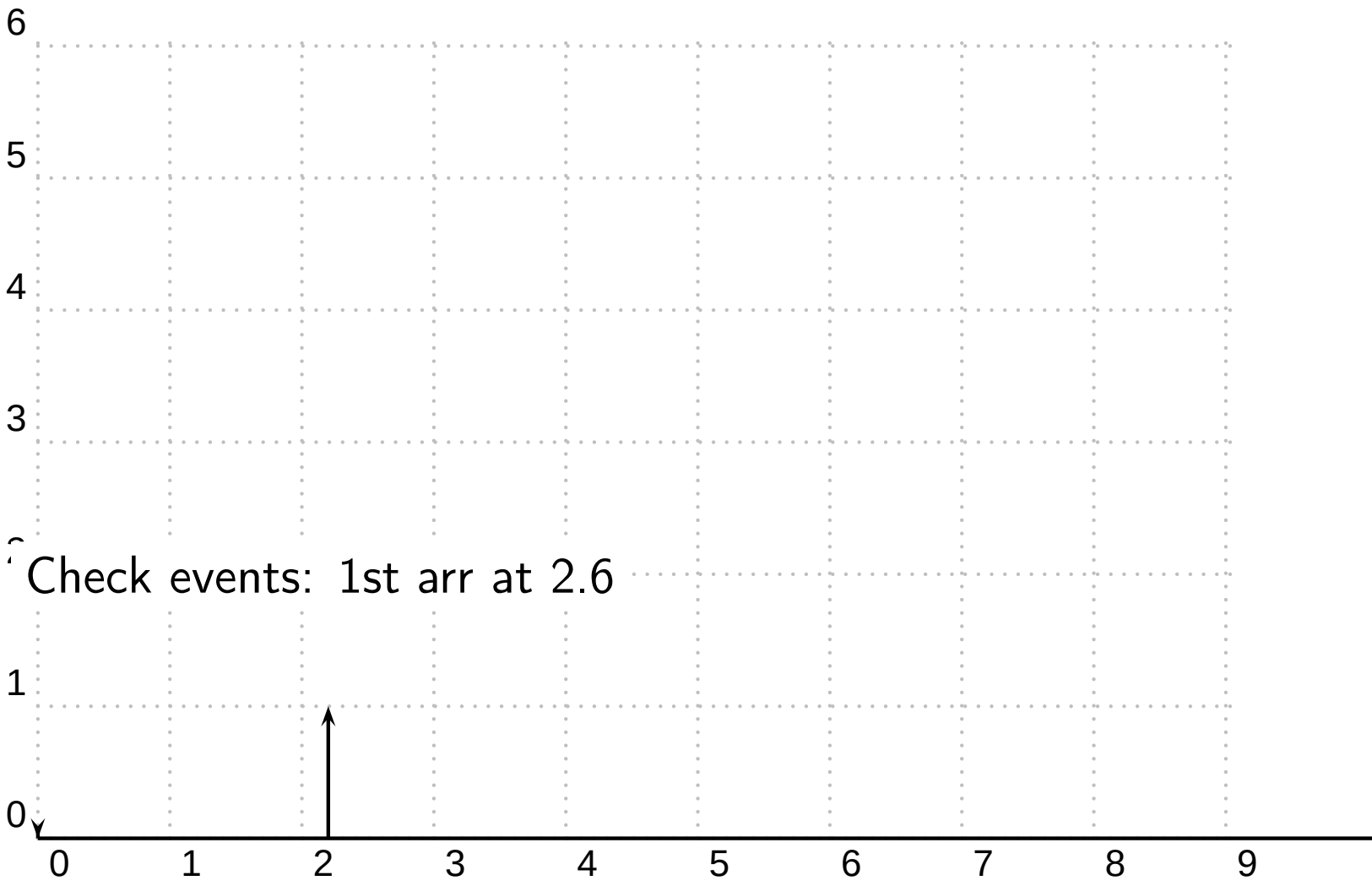
Activity-Oriented Discrete-Event Simulation



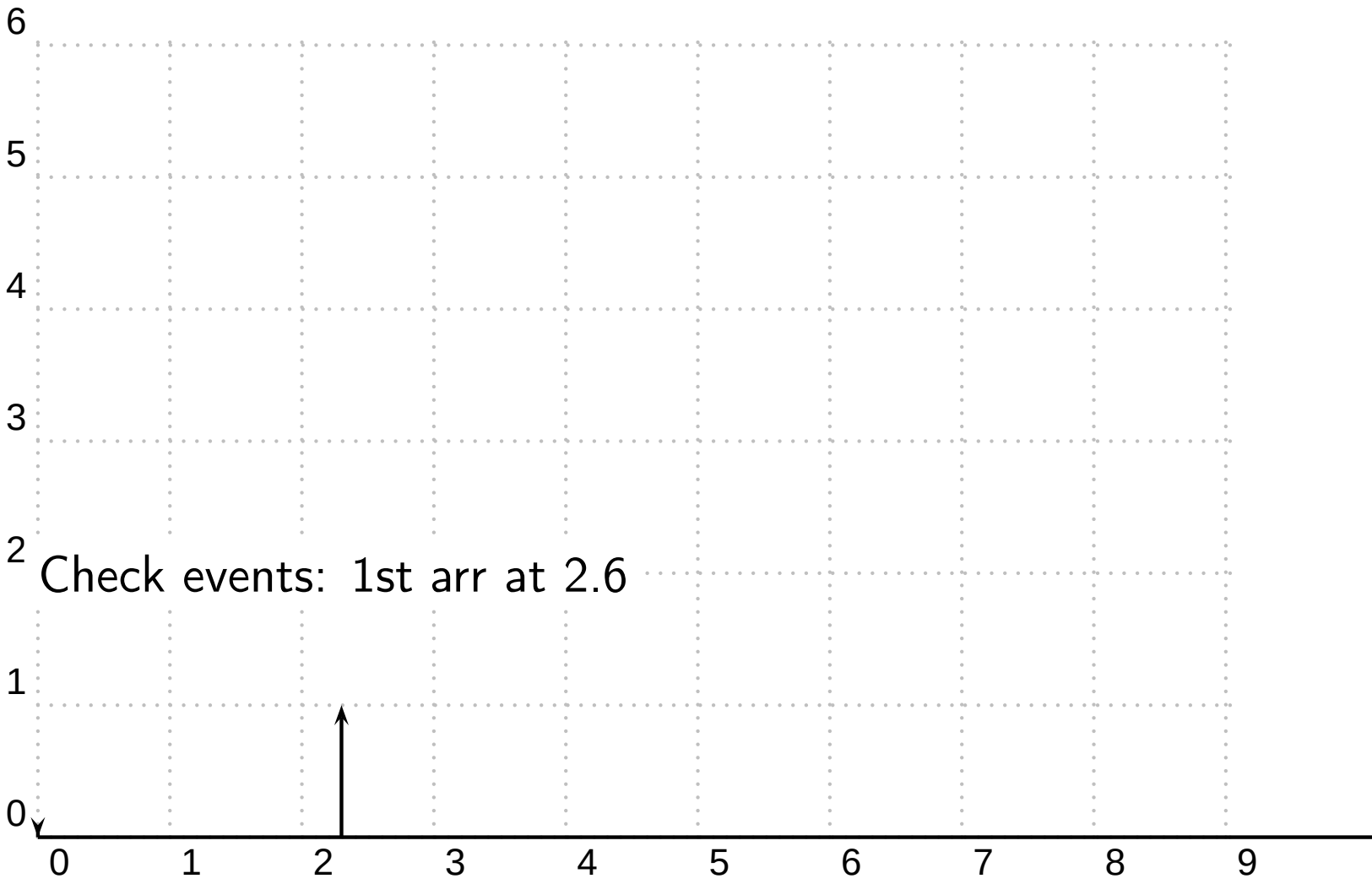
Activity-Oriented Discrete-Event Simulation



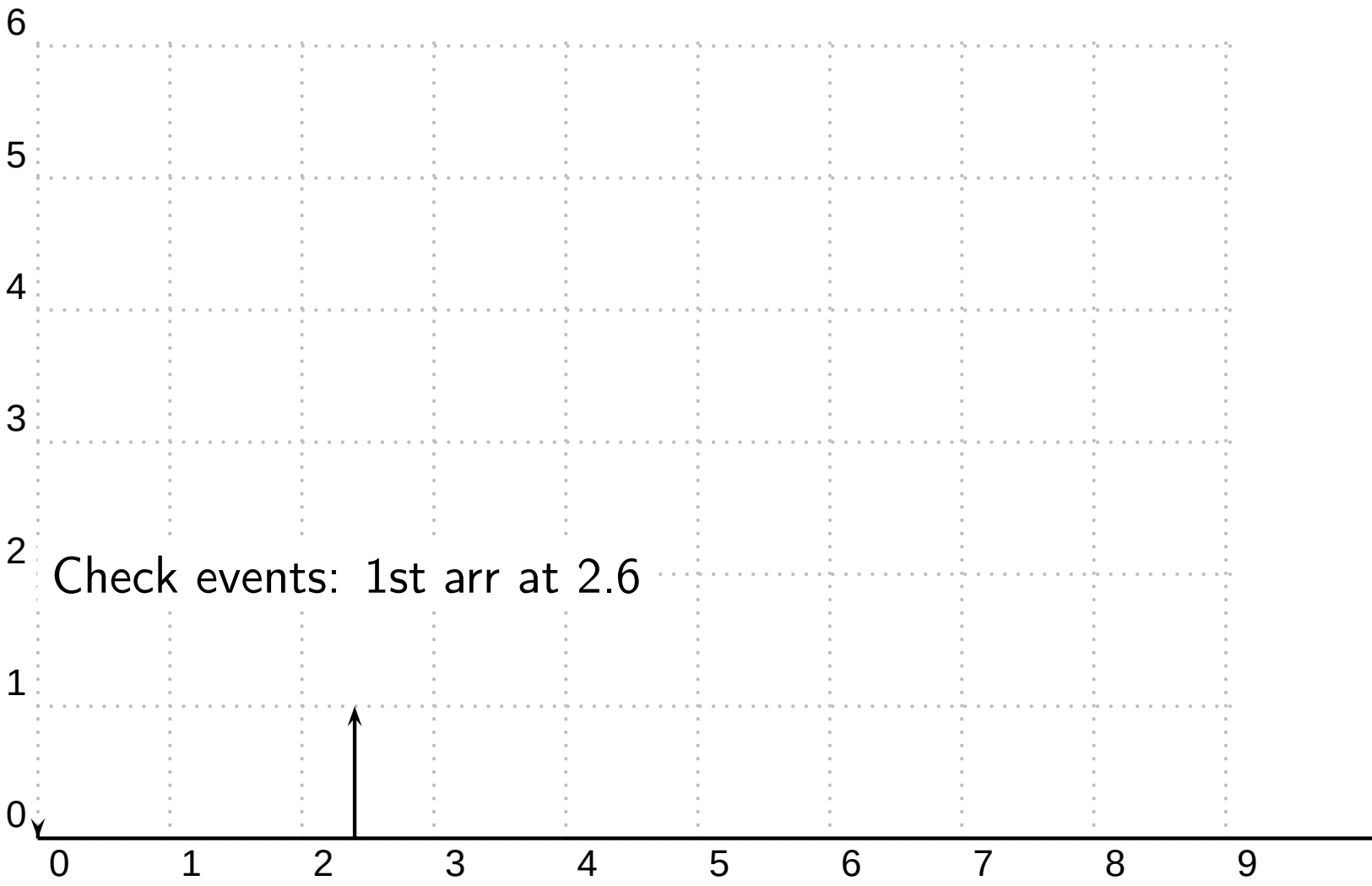
Activity-Oriented Discrete-Event Simulation



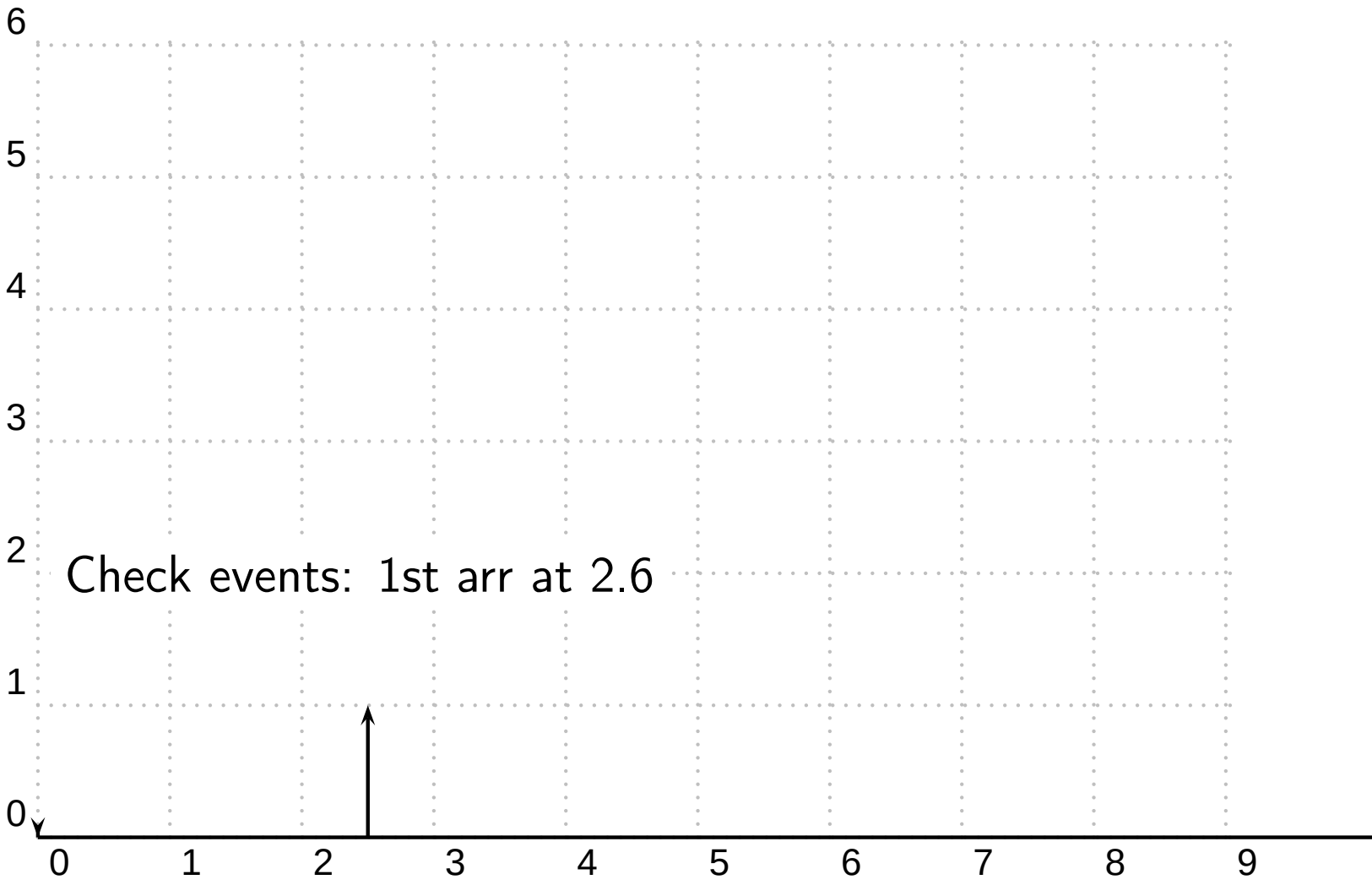
Activity-Oriented Discrete-Event Simulation



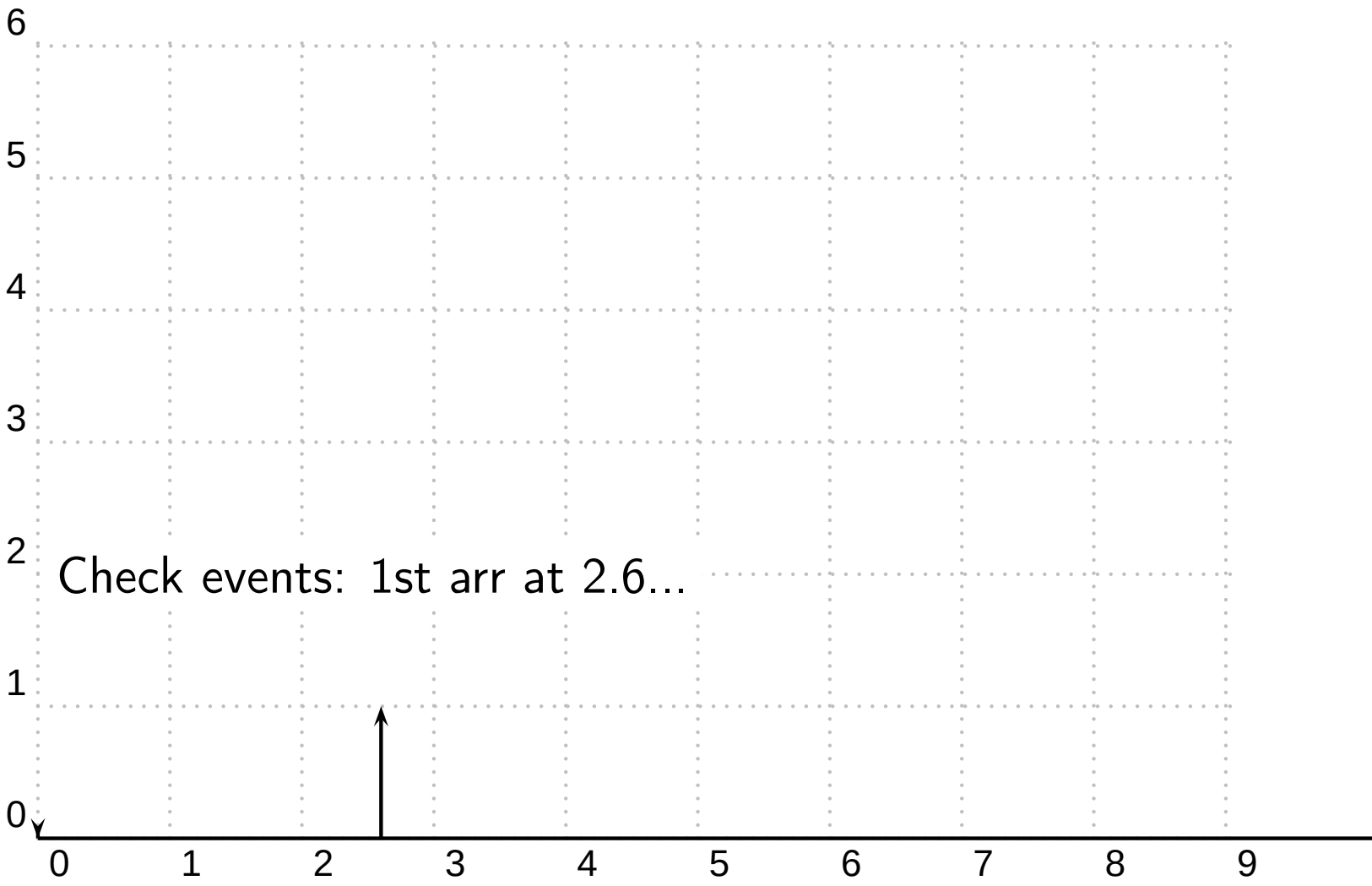
Activity-Oriented Discrete-Event Simulation



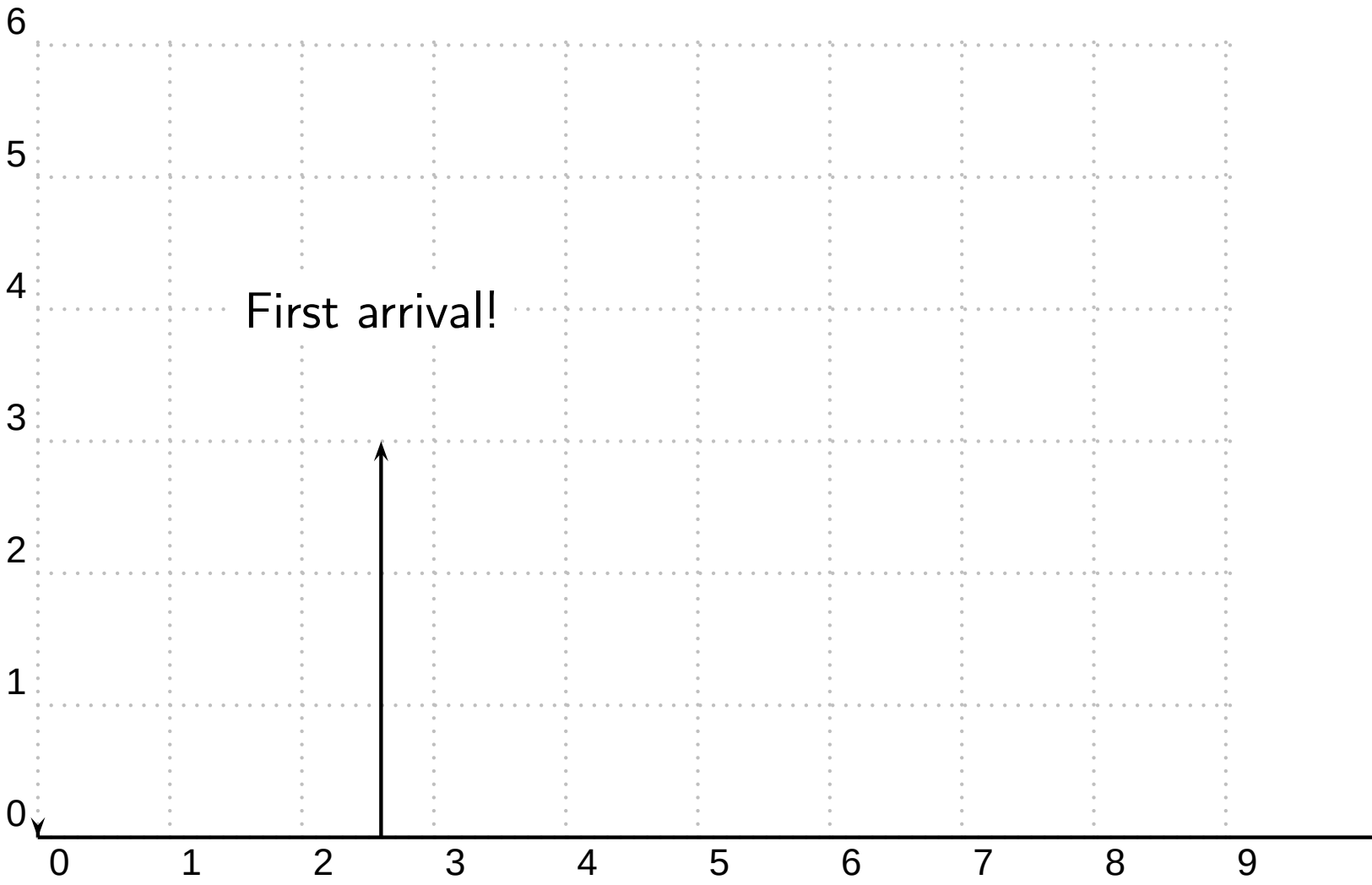
Activity-Oriented Discrete-Event Simulation



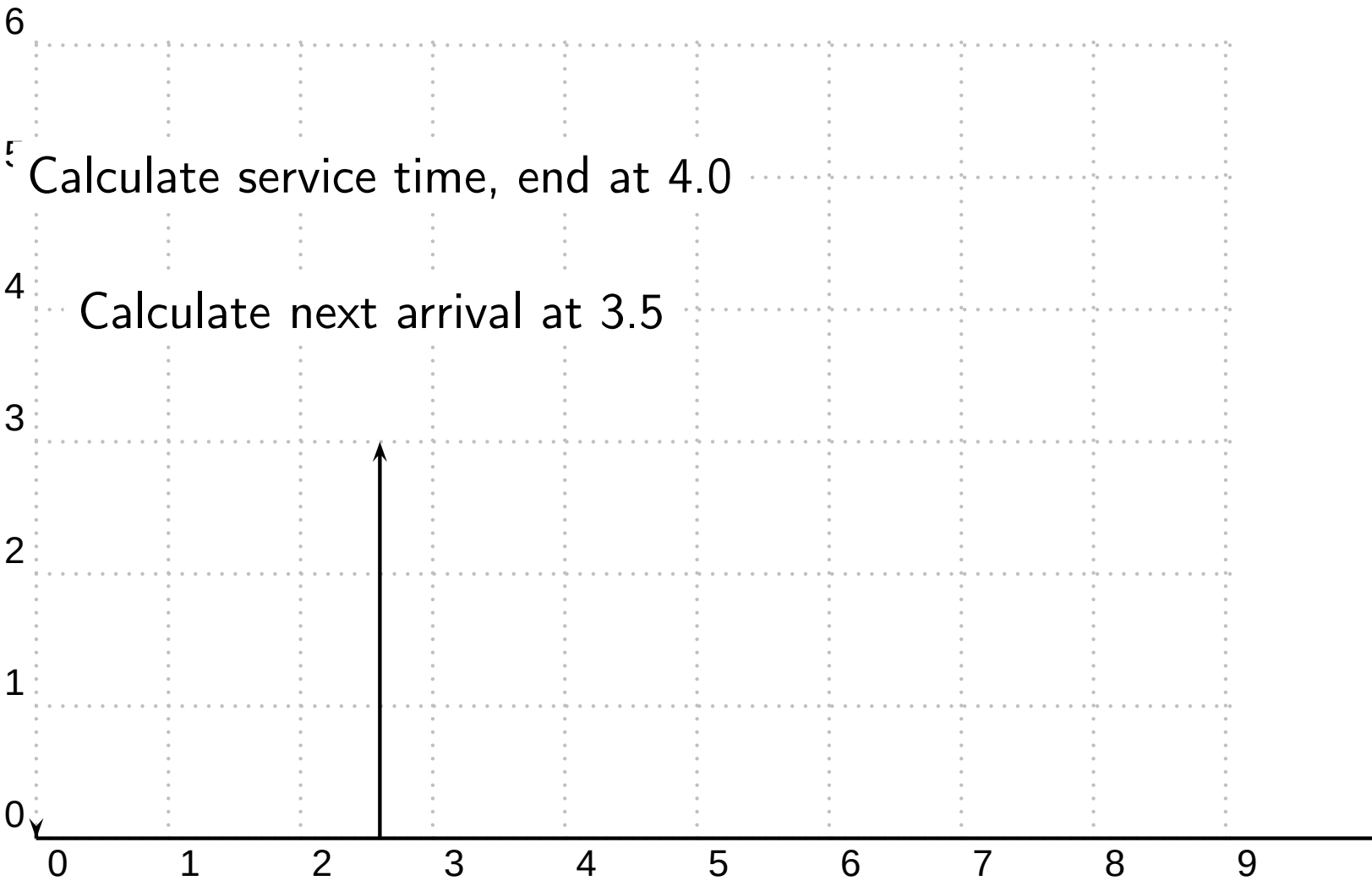
Activity-Oriented Discrete-Event Simulation



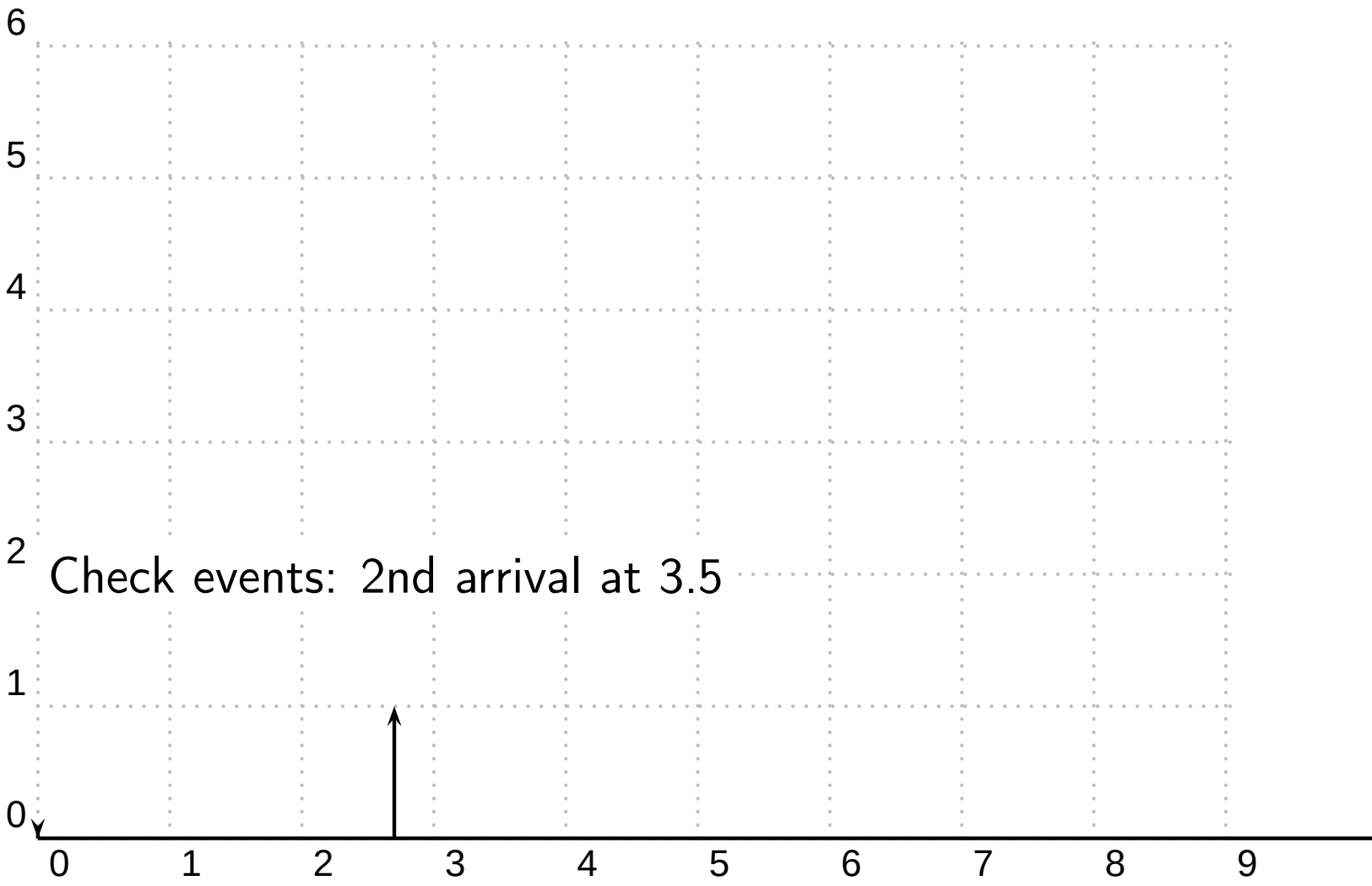
Activity-Oriented Discrete-Event Simulation



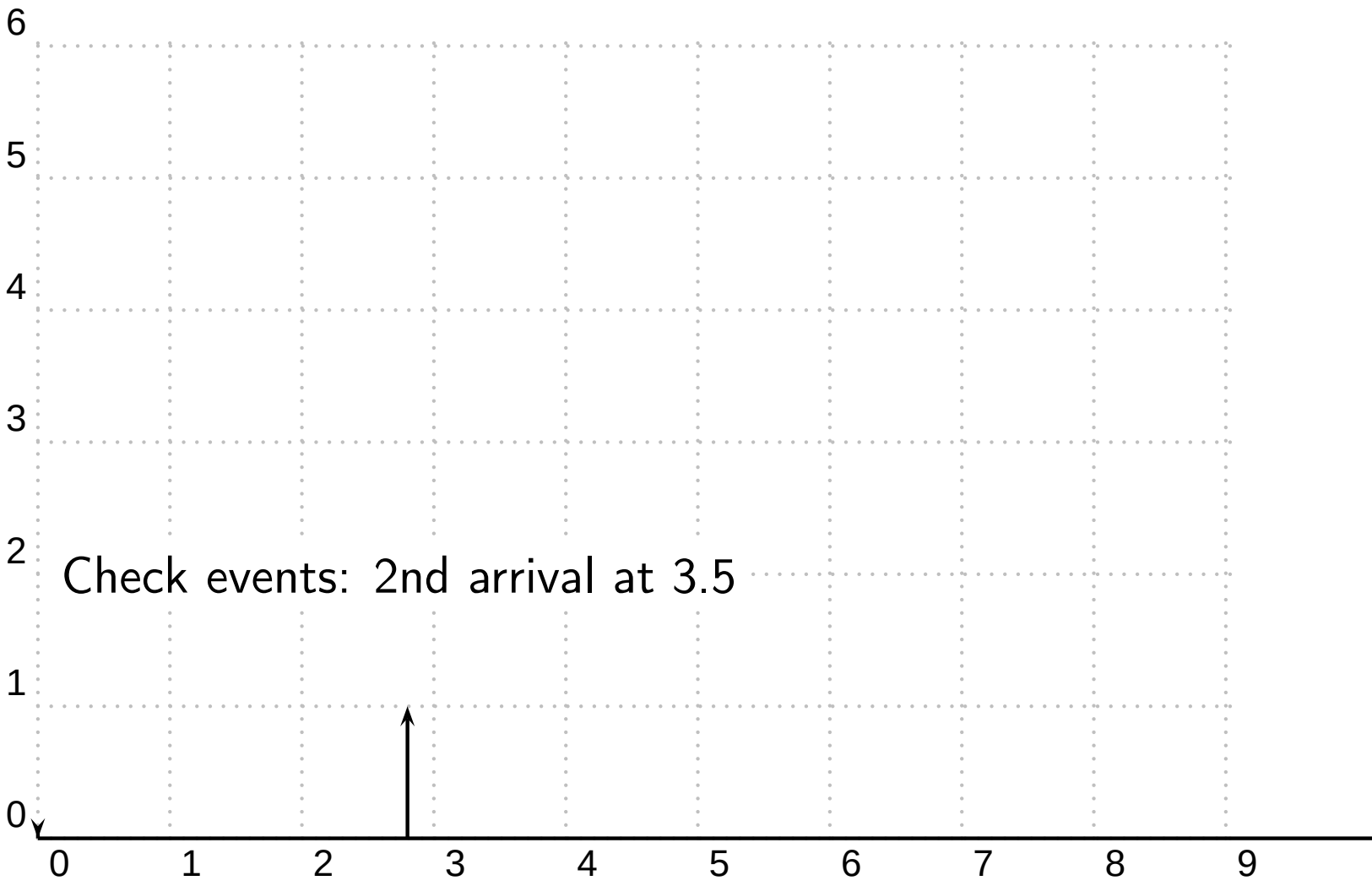
Activity-Oriented Discrete-Event Simulation



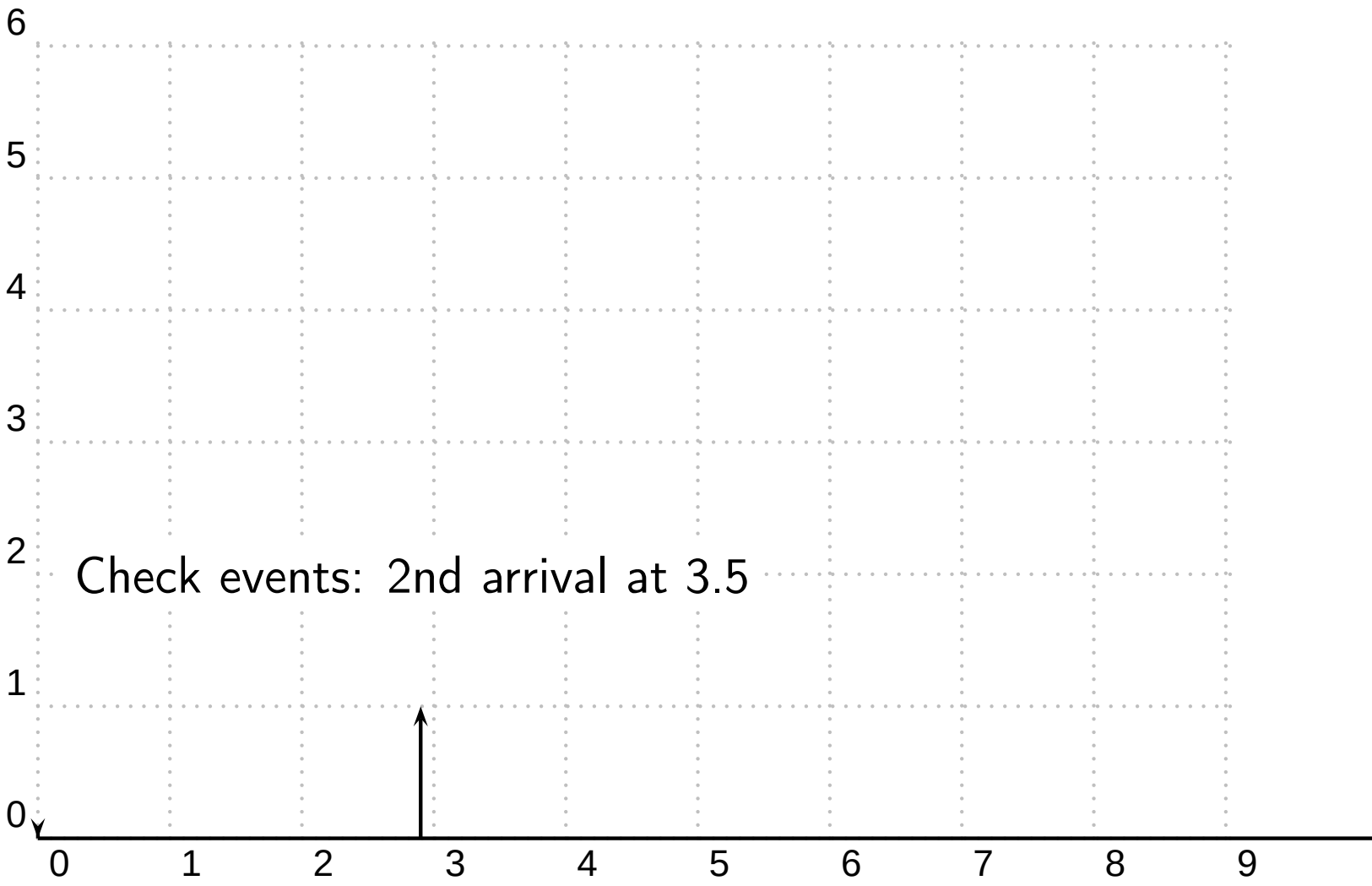
Activity-Oriented Discrete-Event Simulation



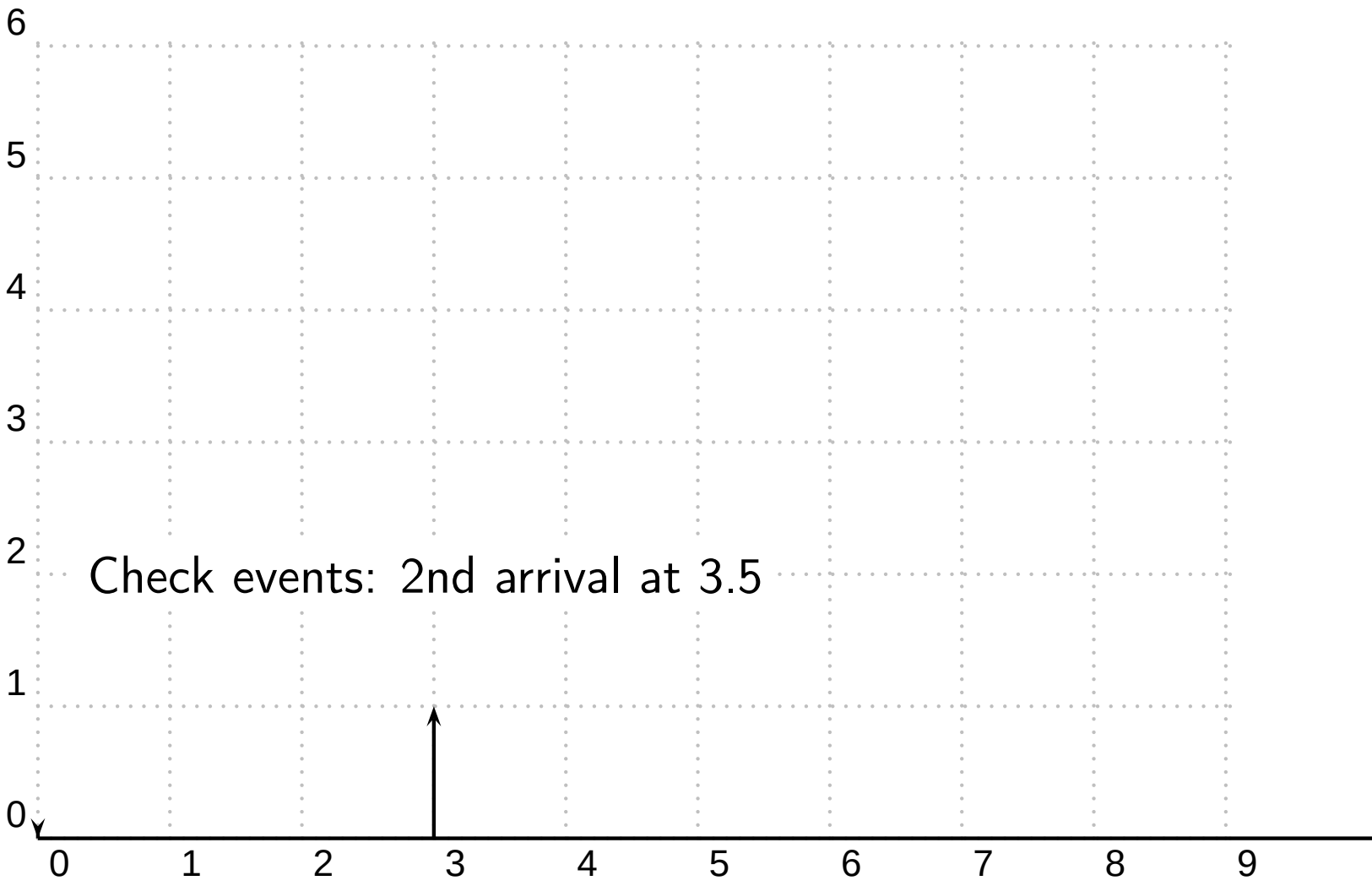
Activity-Oriented Discrete-Event Simulation



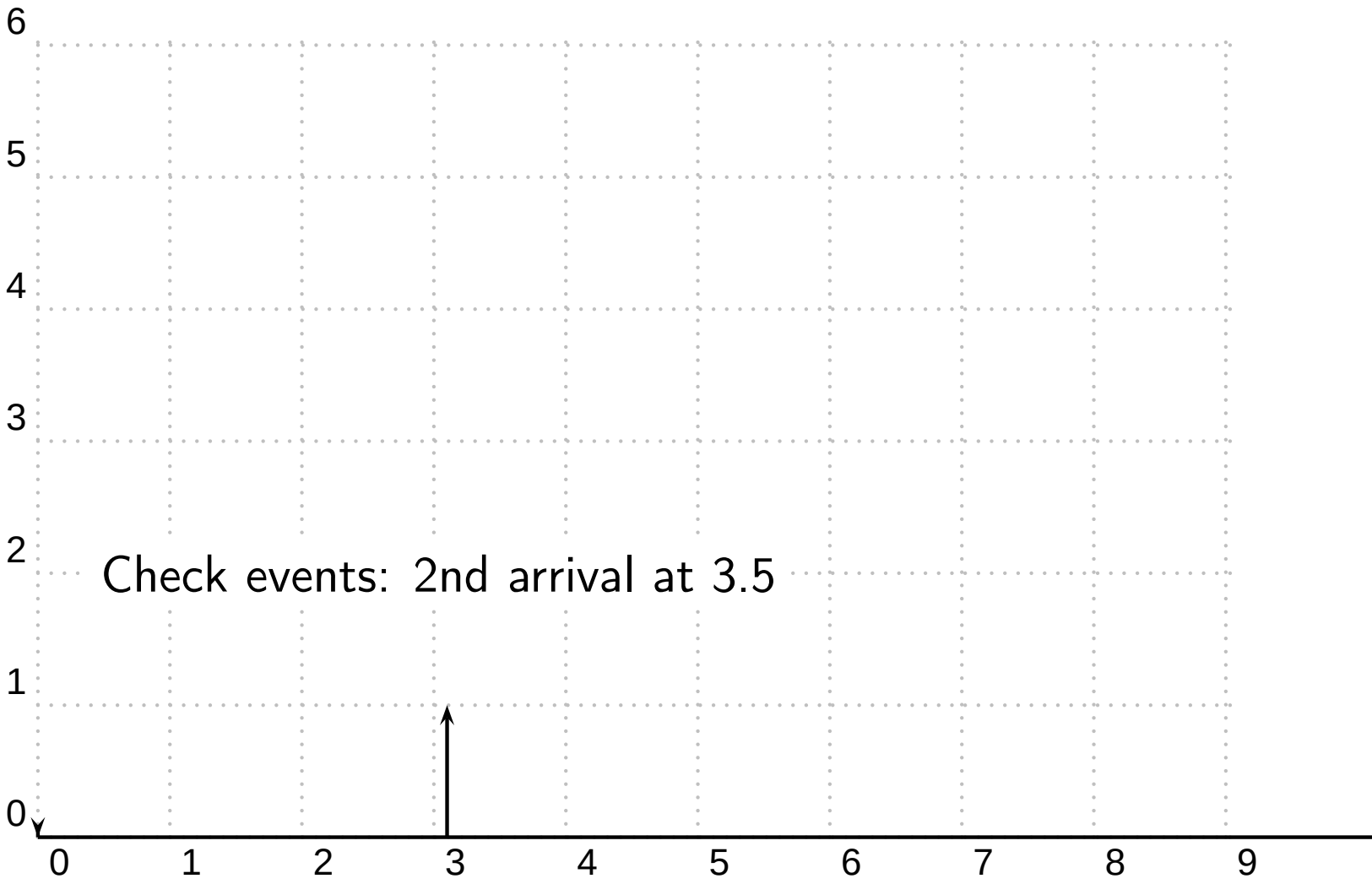
Activity-Oriented Discrete-Event Simulation



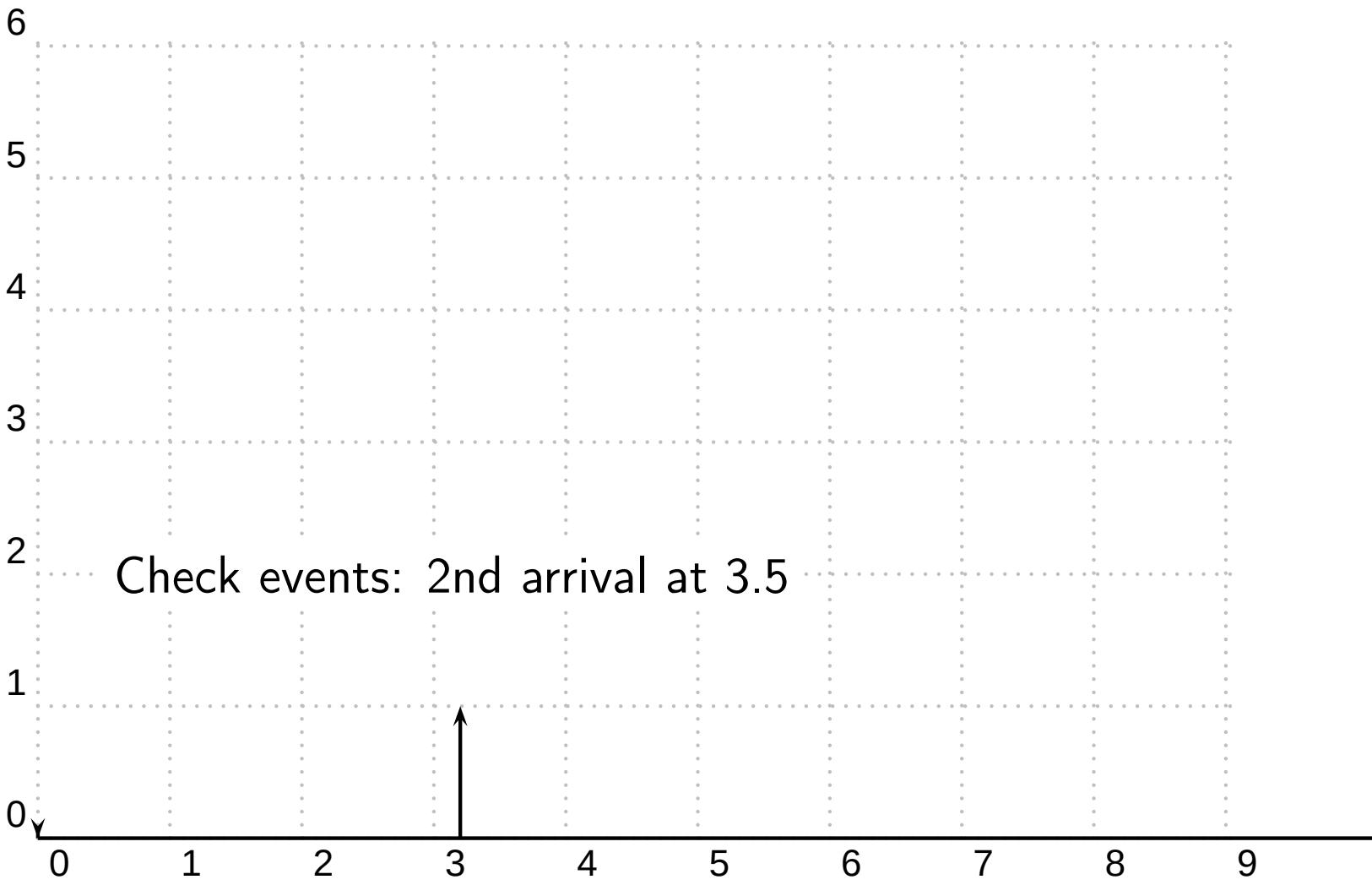
Activity-Oriented Discrete-Event Simulation



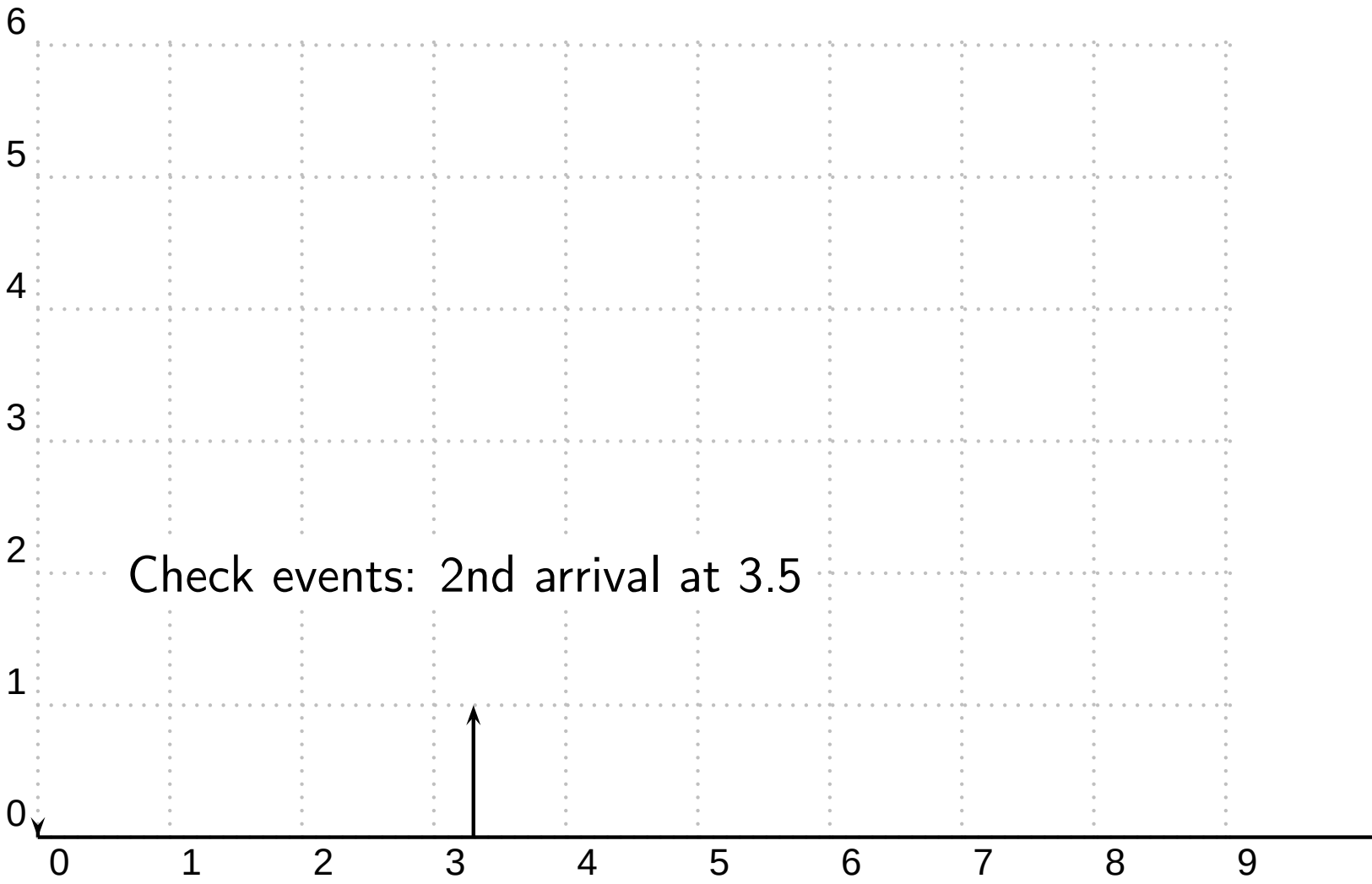
Activity-Oriented Discrete-Event Simulation



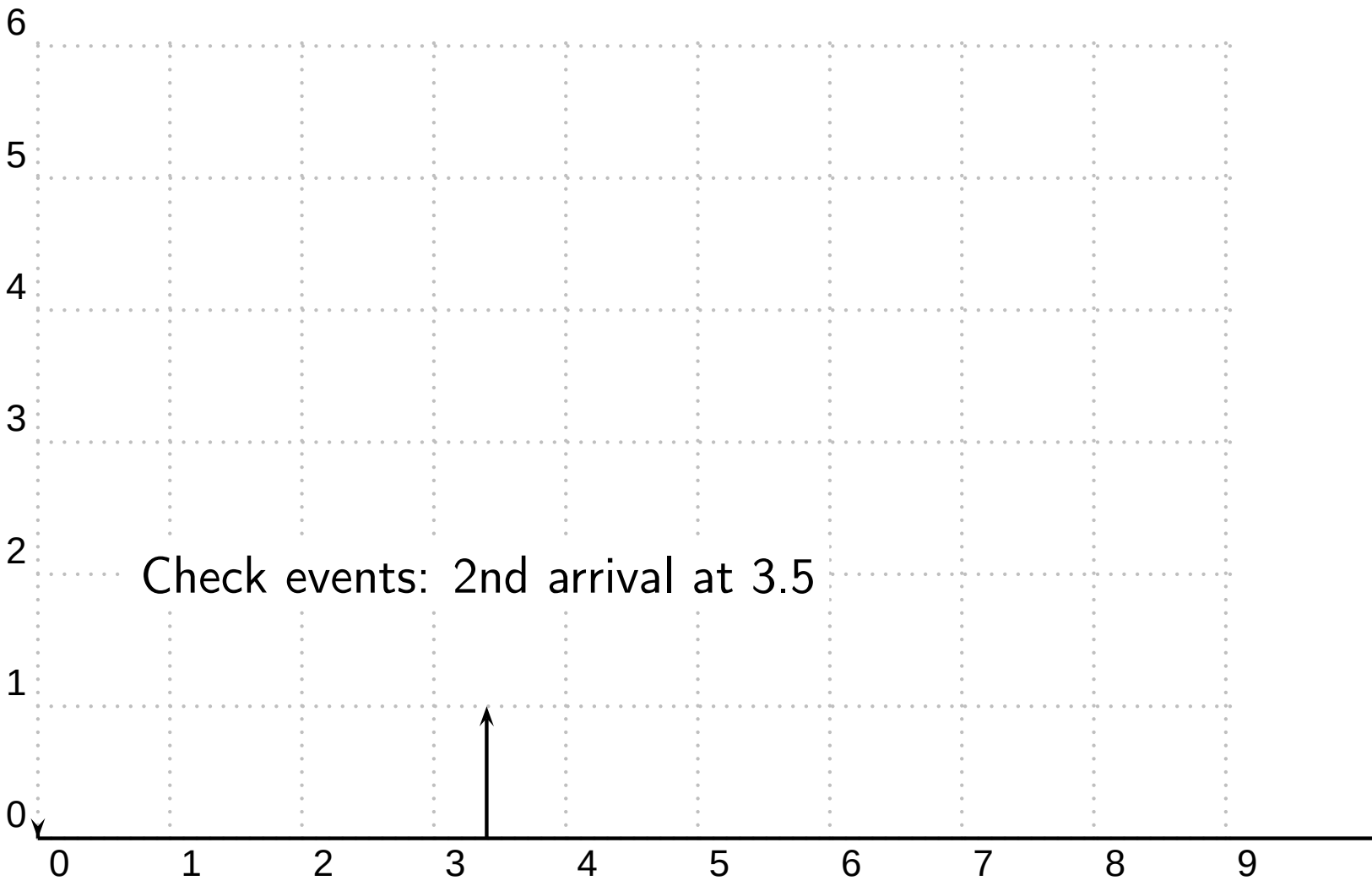
Activity-Oriented Discrete-Event Simulation



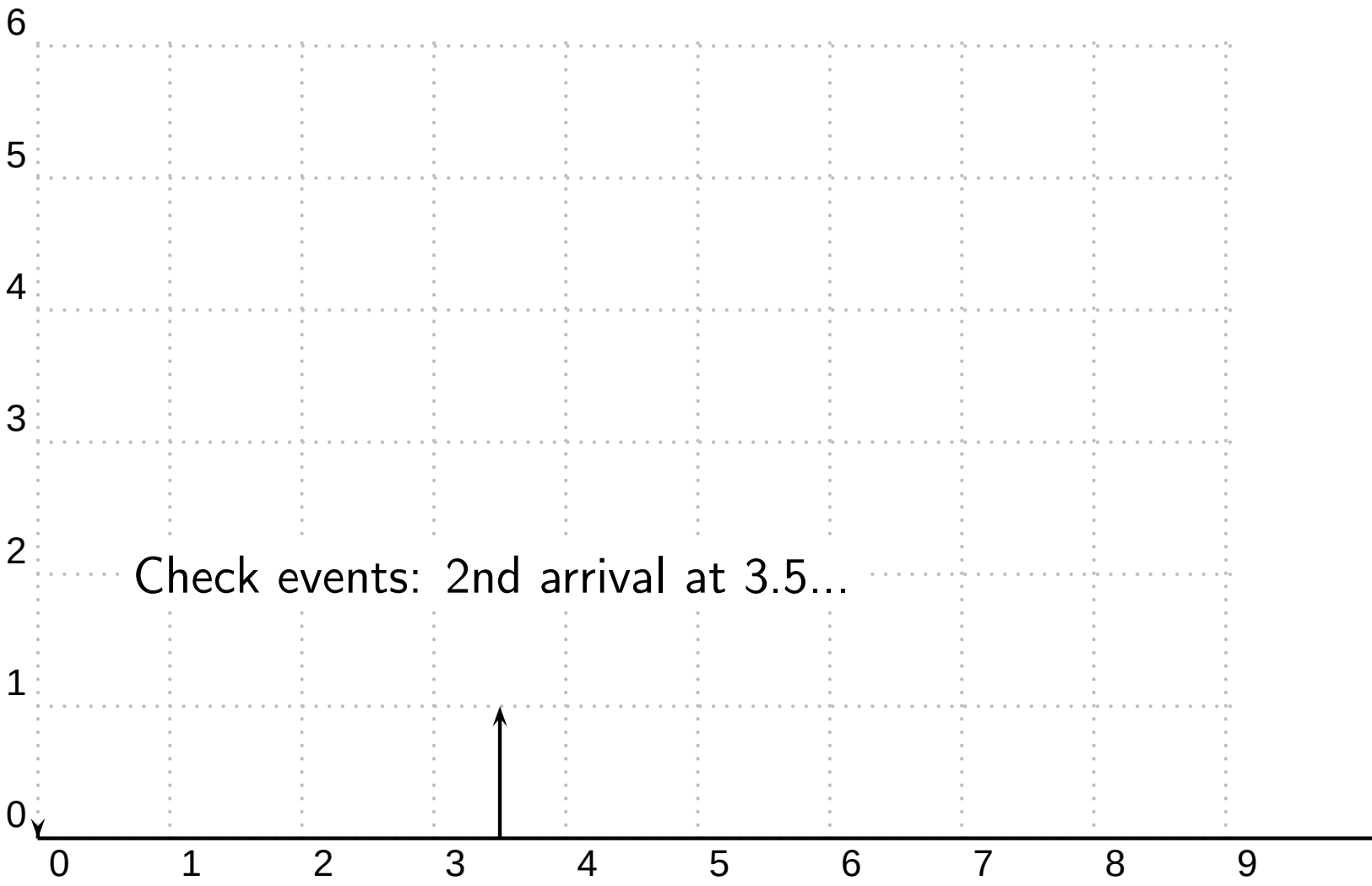
Activity-Oriented Discrete-Event Simulation



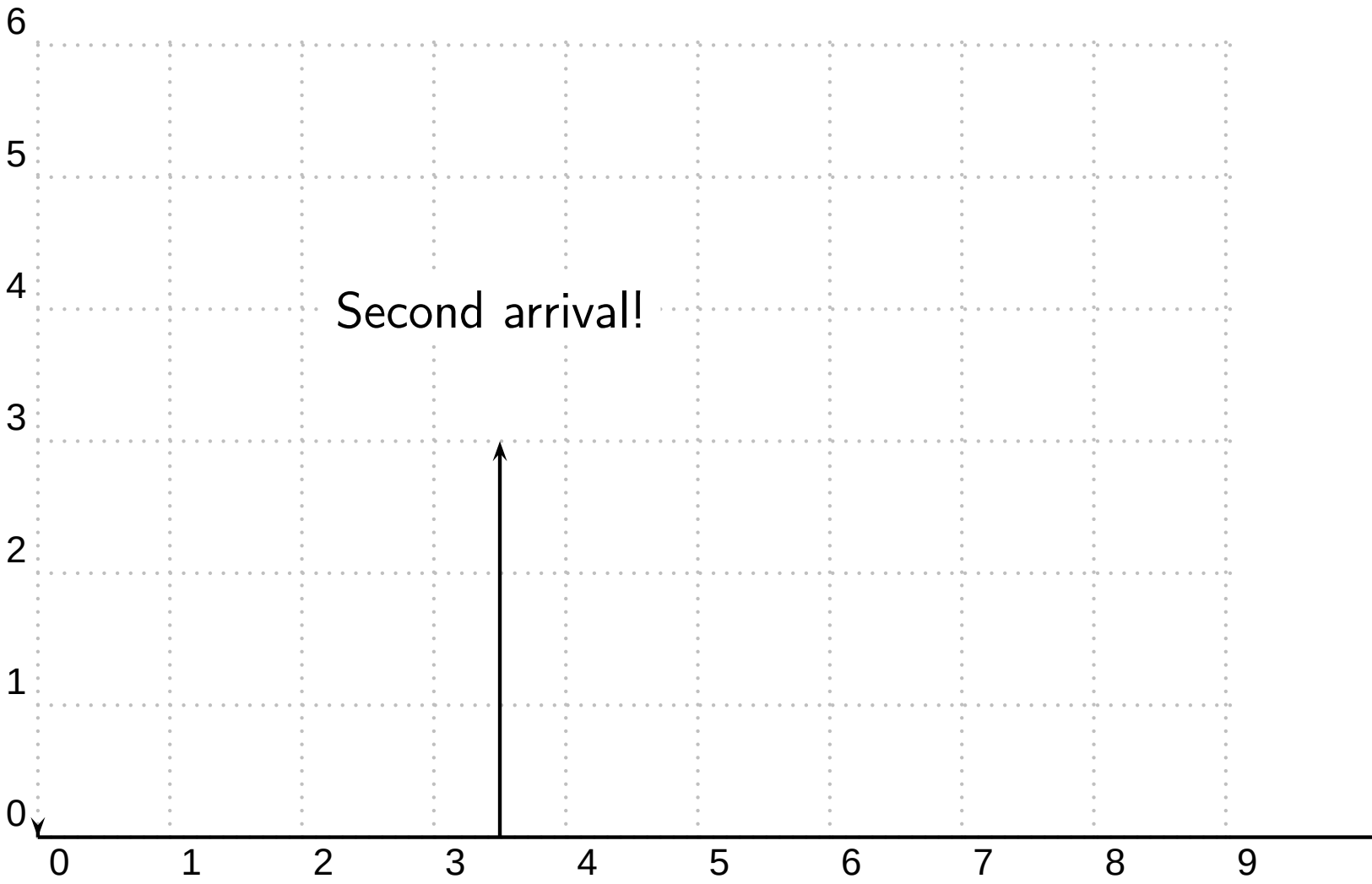
Activity-Oriented Discrete-Event Simulation



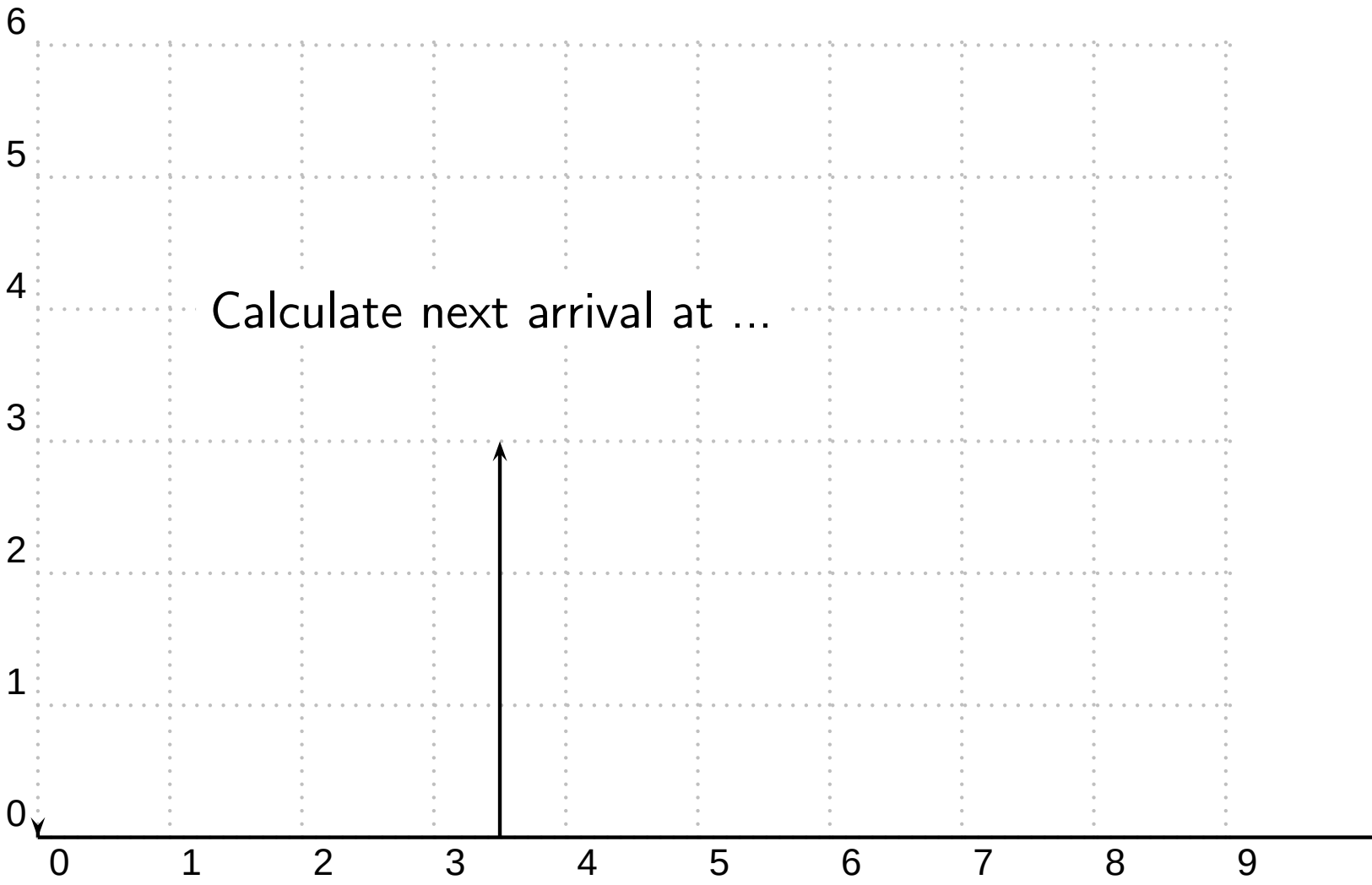
Activity-Oriented Discrete-Event Simulation



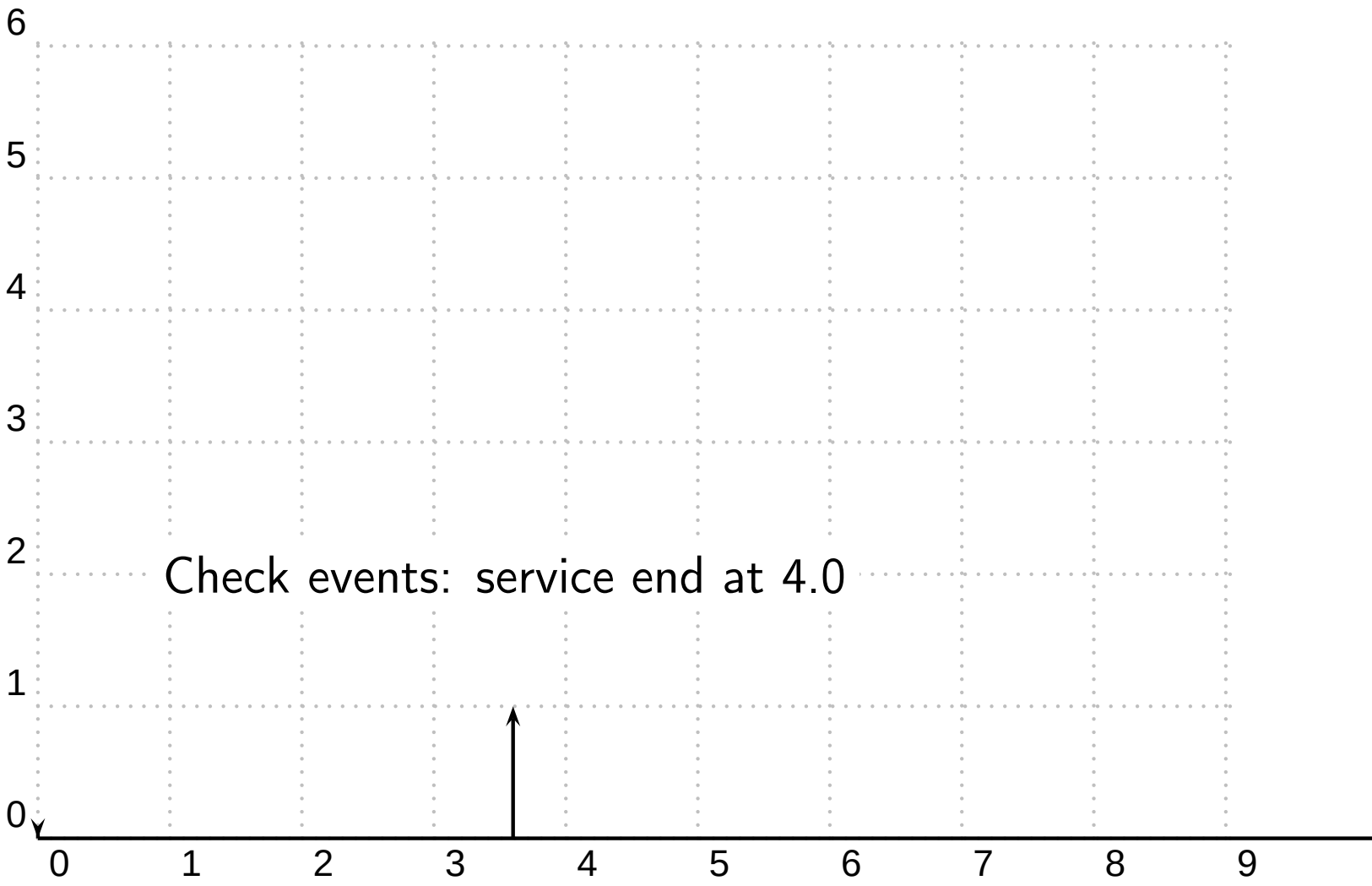
Activity-Oriented Discrete-Event Simulation



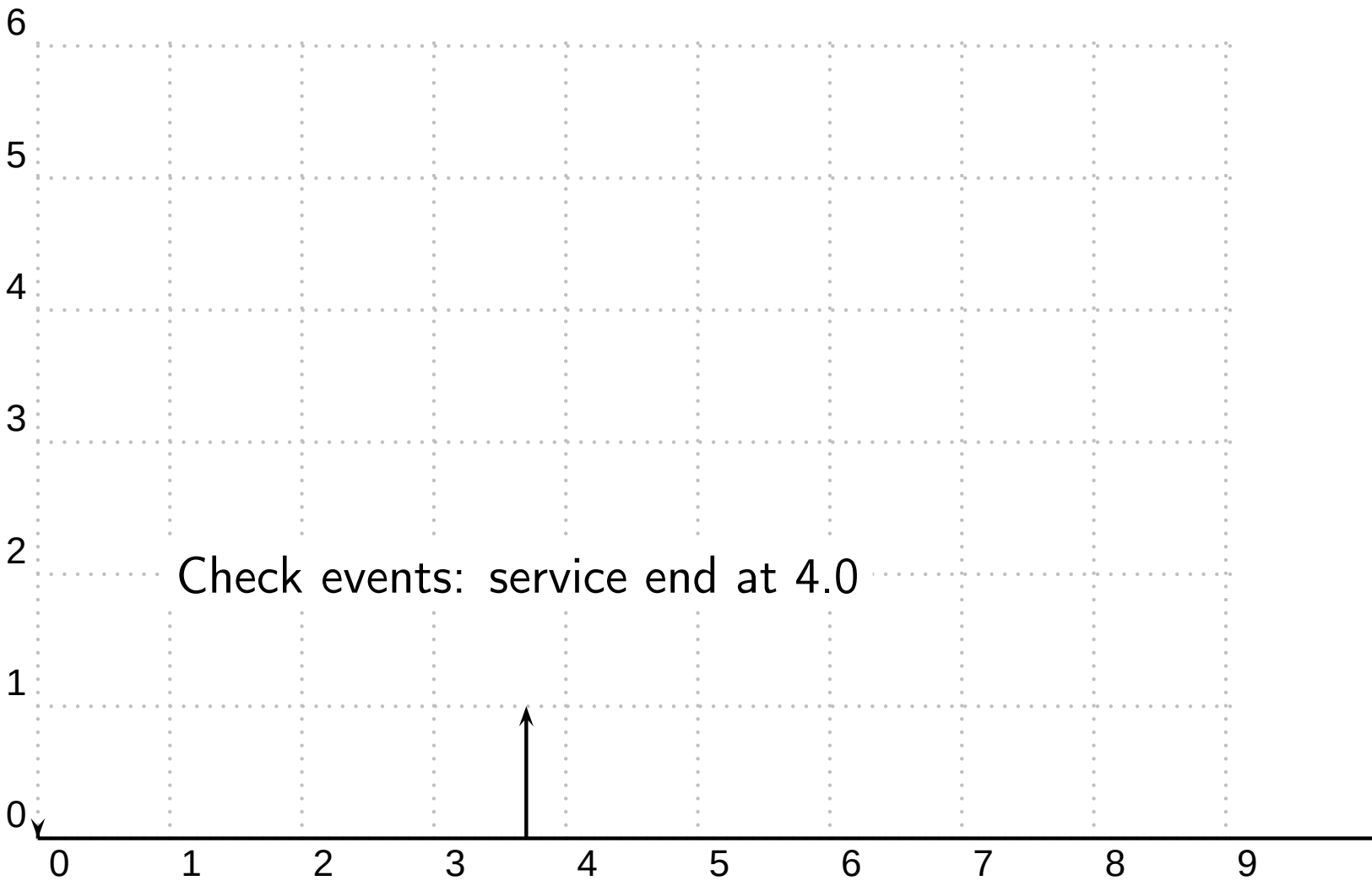
Activity-Oriented Discrete-Event Simulation



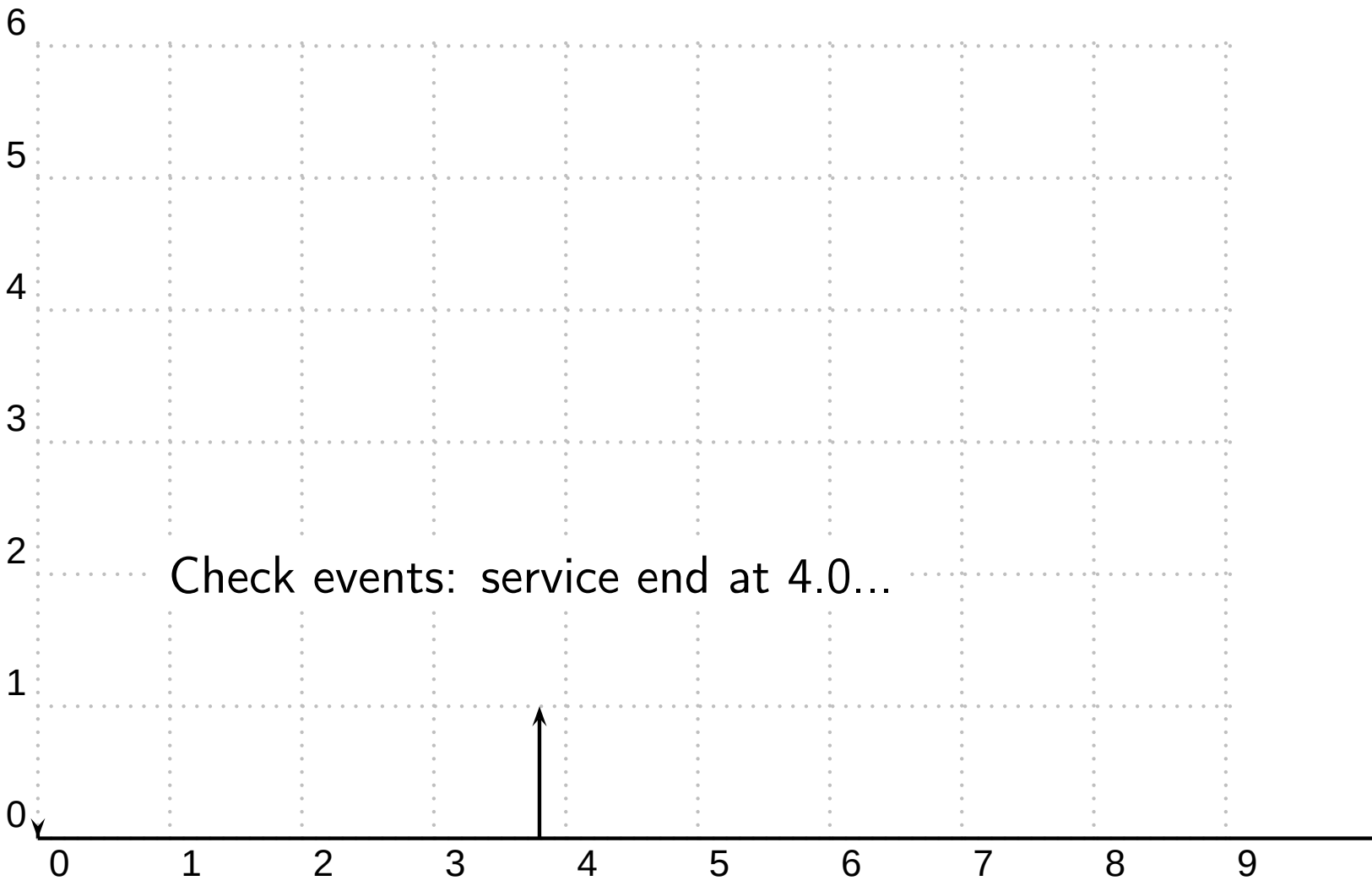
Activity-Oriented Discrete-Event Simulation



Activity-Oriented Discrete-Event Simulation



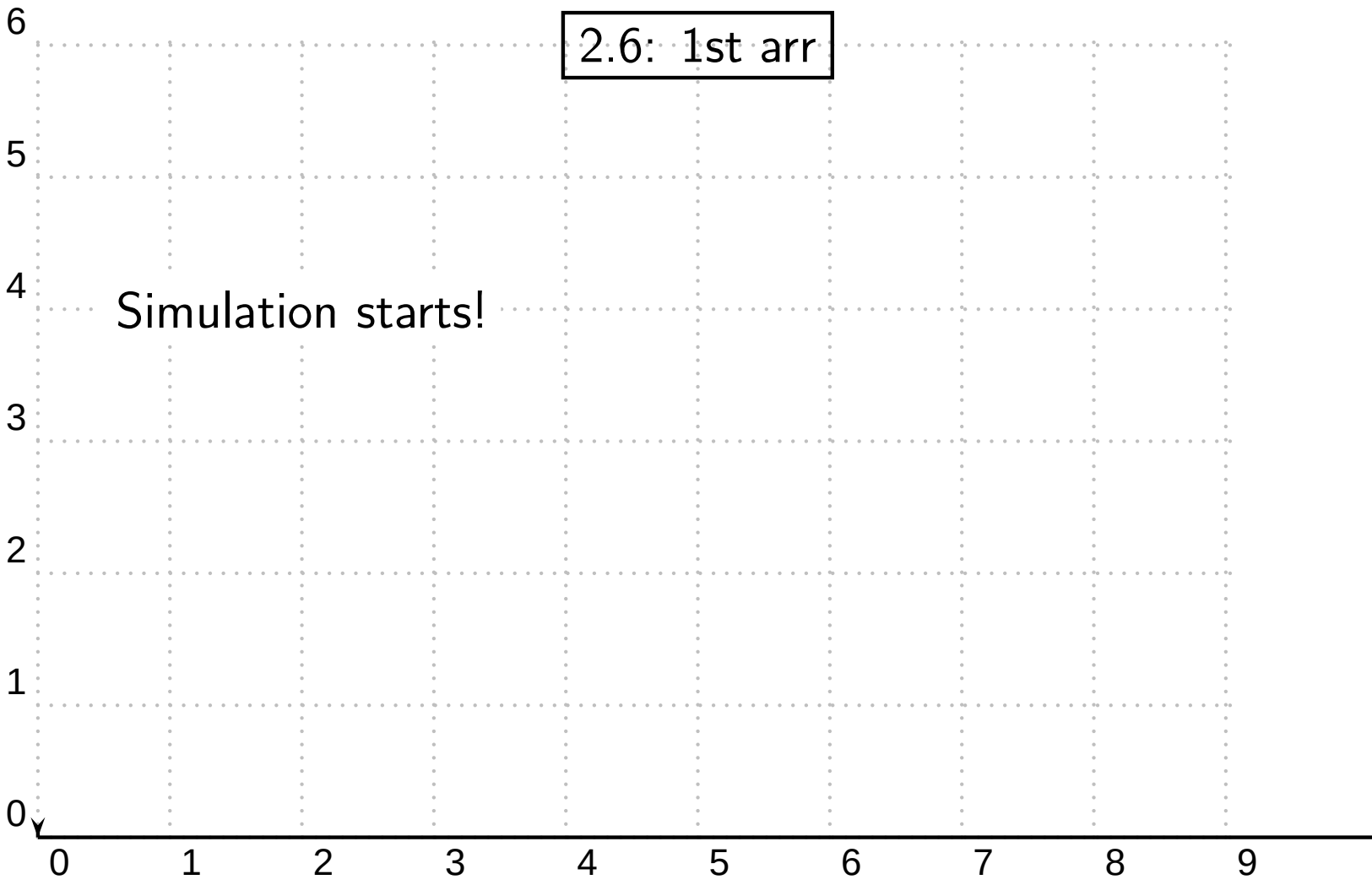
Activity-Oriented Discrete-Event Simulation



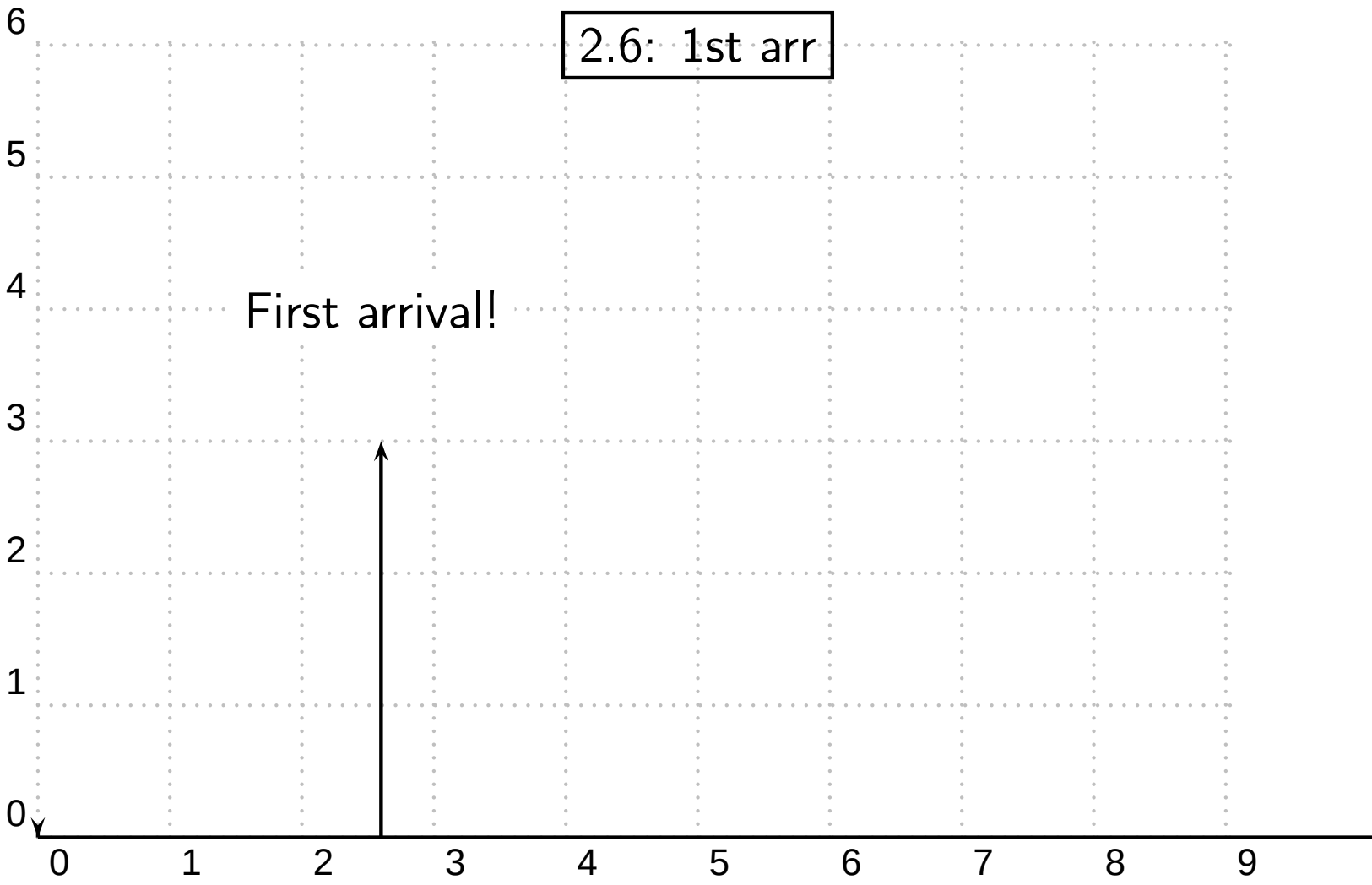
Discrete-Event Simulation World Views

- Activity-oriented
  - fixed increment of time
  - time-consuming
- Event-oriented
  - on each event, generate next event and put into event queue
  - simulation time advances to next event
  - faster than activity-oriented
- Process-oriented
  - abstract one object into a process
  - easier to maintain in the end

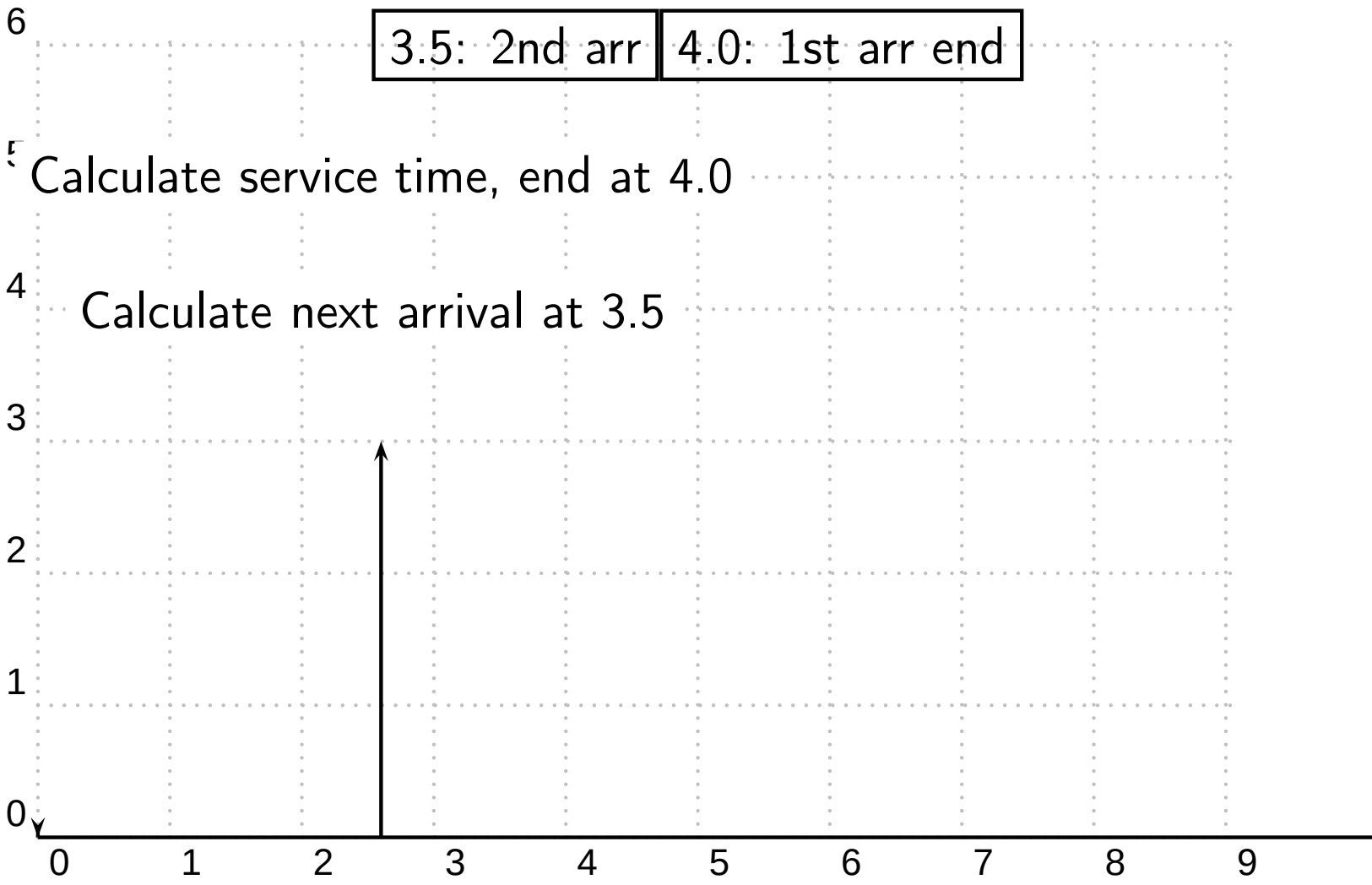
Event-Oriented Discrete-Event Simulation



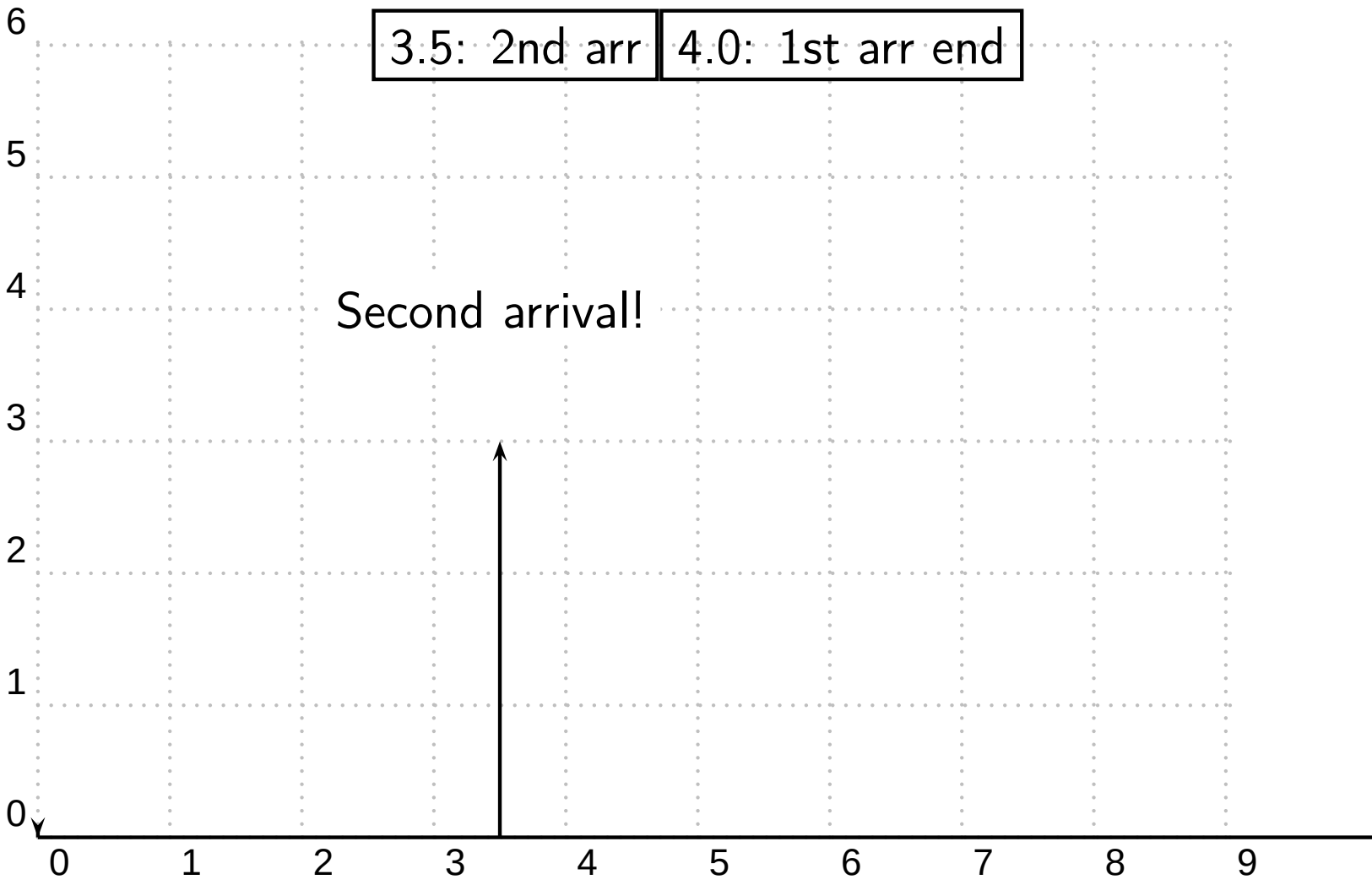
Event-Oriented Discrete-Event Simulation



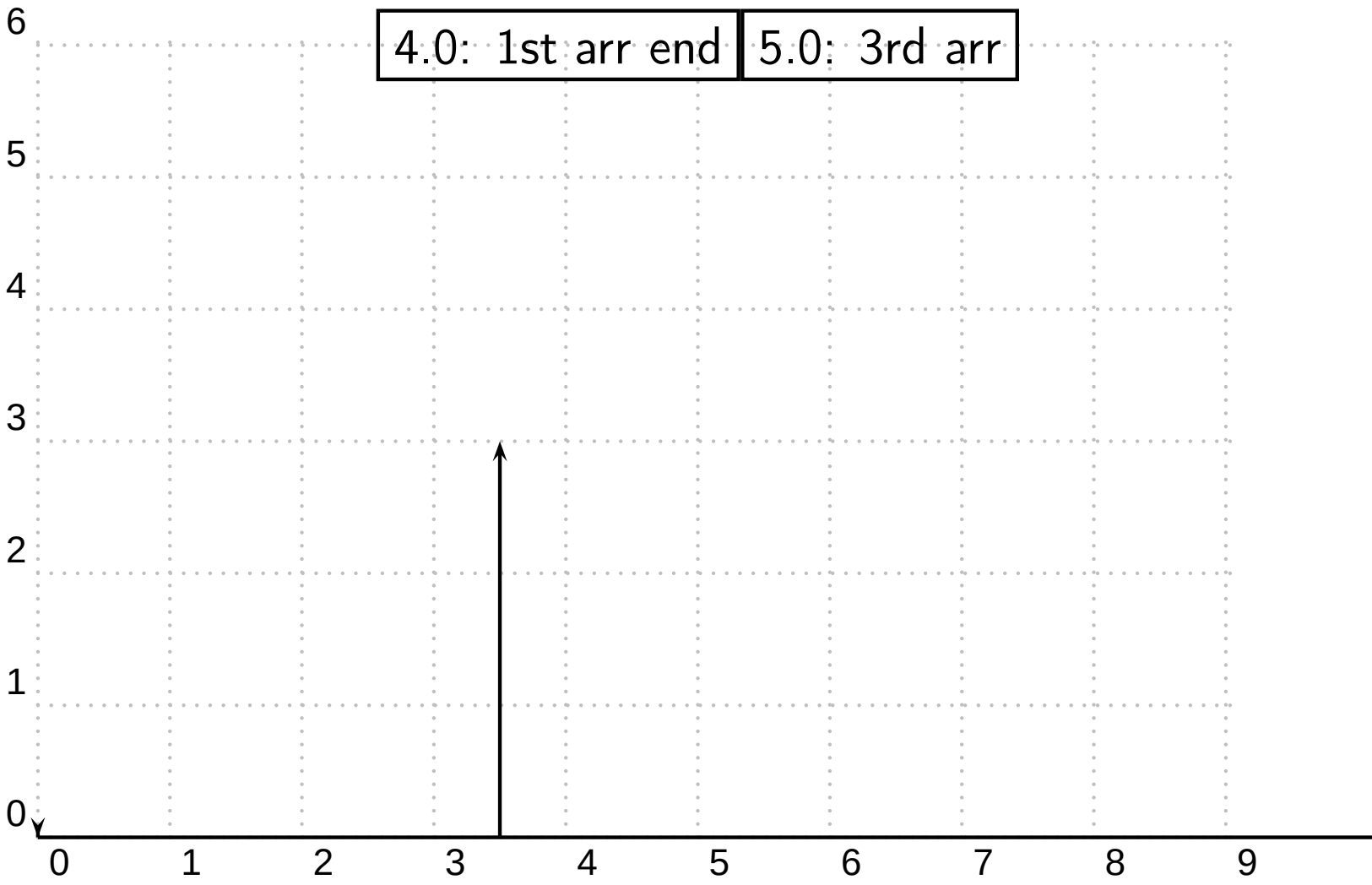
Event-Oriented Discrete-Event Simulation



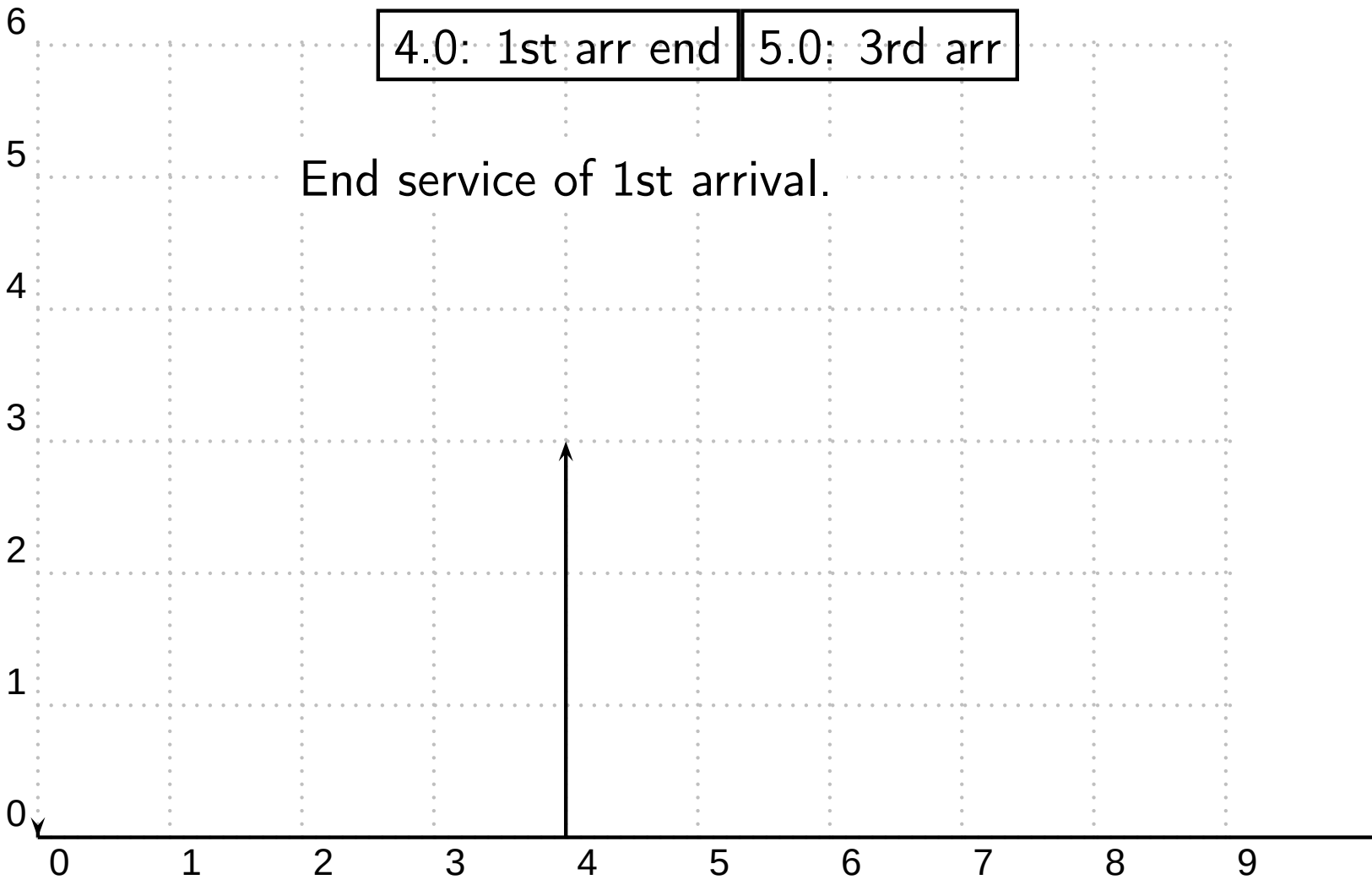
Event-Oriented Discrete-Event Simulation



Event-Oriented Discrete-Event Simulation



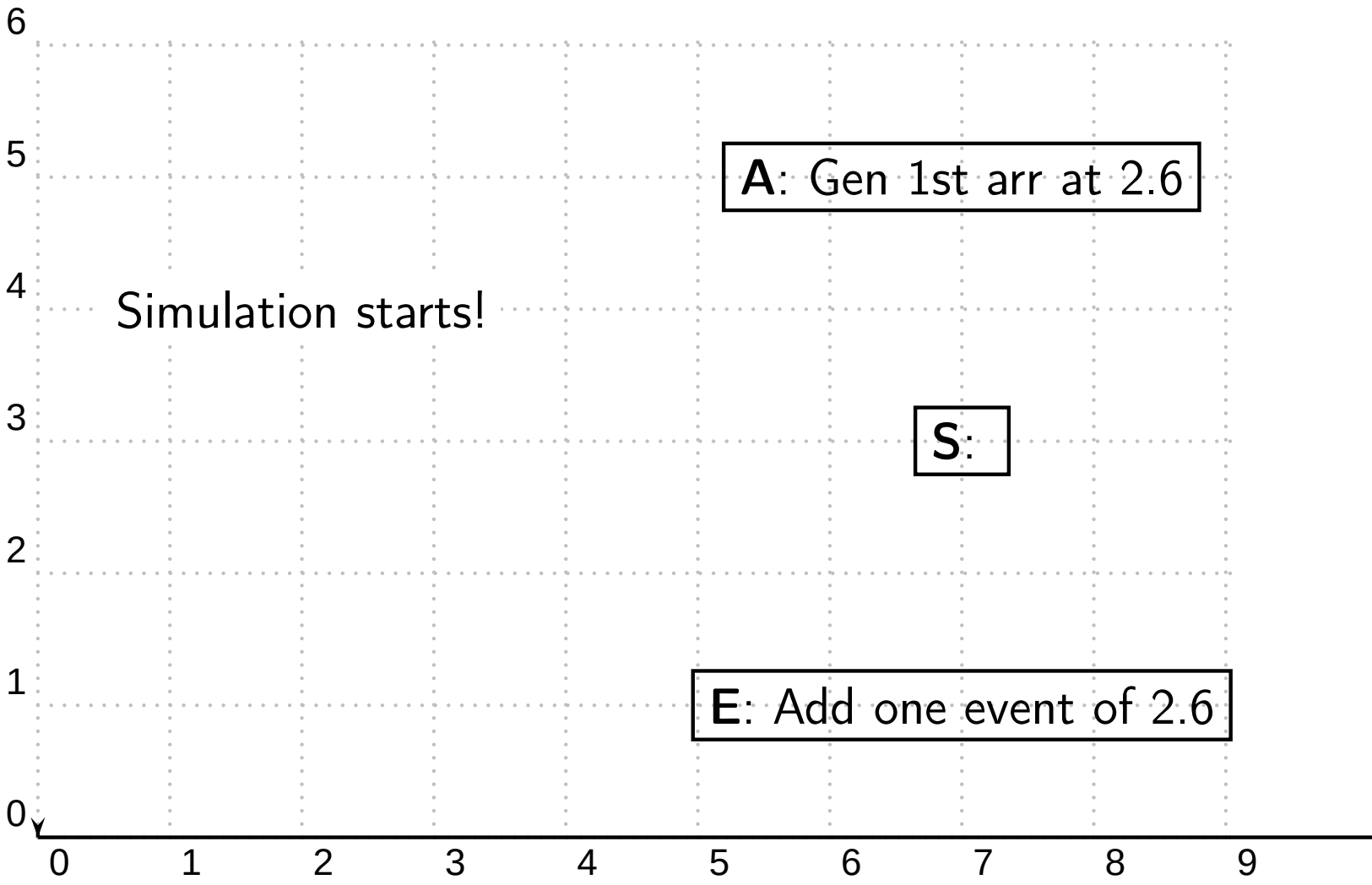
Event-Oriented Discrete-Event Simulation



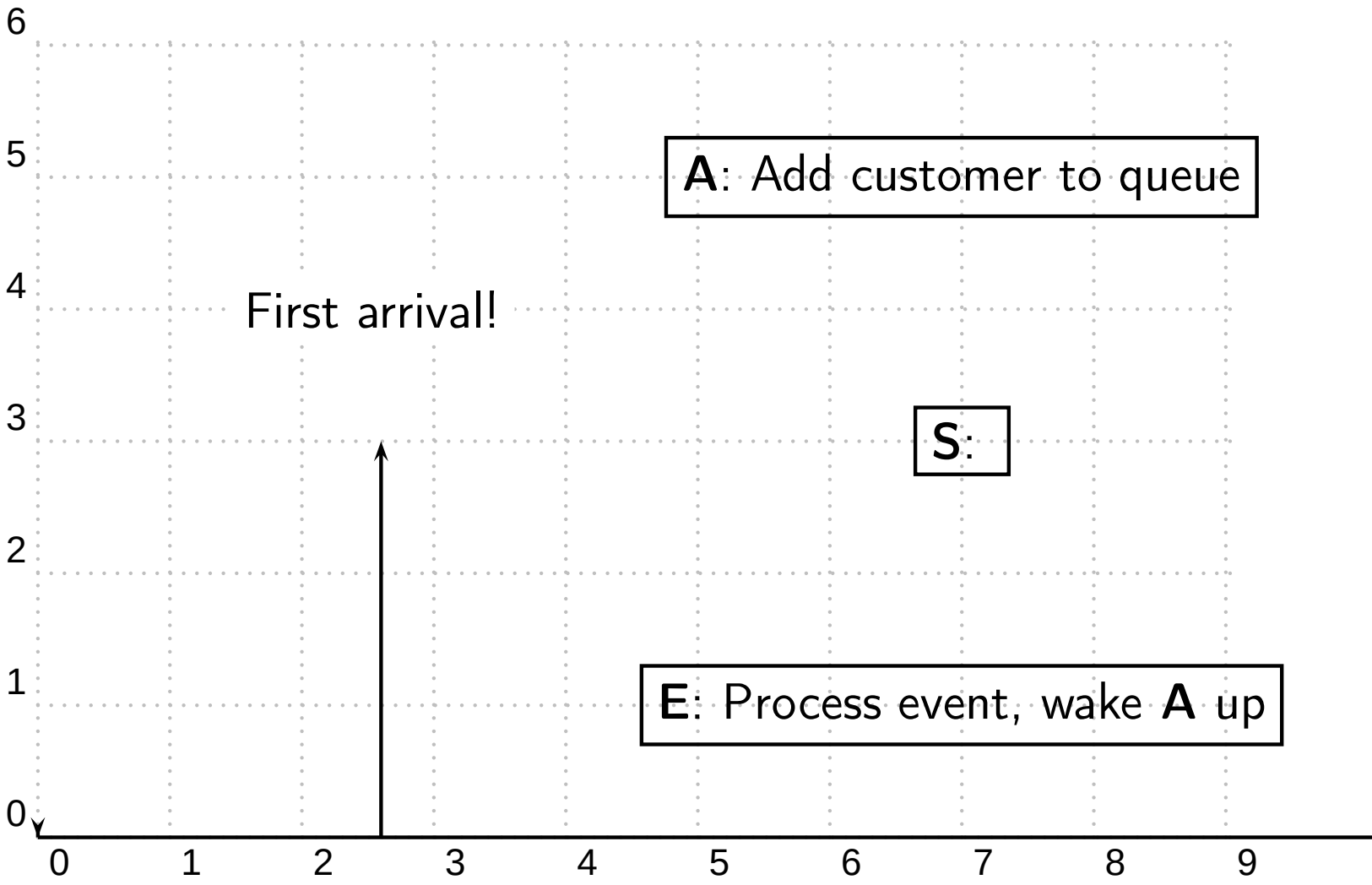
Discrete-Event Simulation World Views

- Activity-oriented
  - fixed increment of time
  - time-consuming
- Event-oriented
  - on each event, generate next event and put into event queue
  - simulation time advances to next event
  - faster than activity-oriented
- Process-oriented
  - abstract one object into a process
    - Arrival process for customers, or **A**
    - Clerk process, or **S**
    - Event manager, or **E**
  - easier to maintain in the end

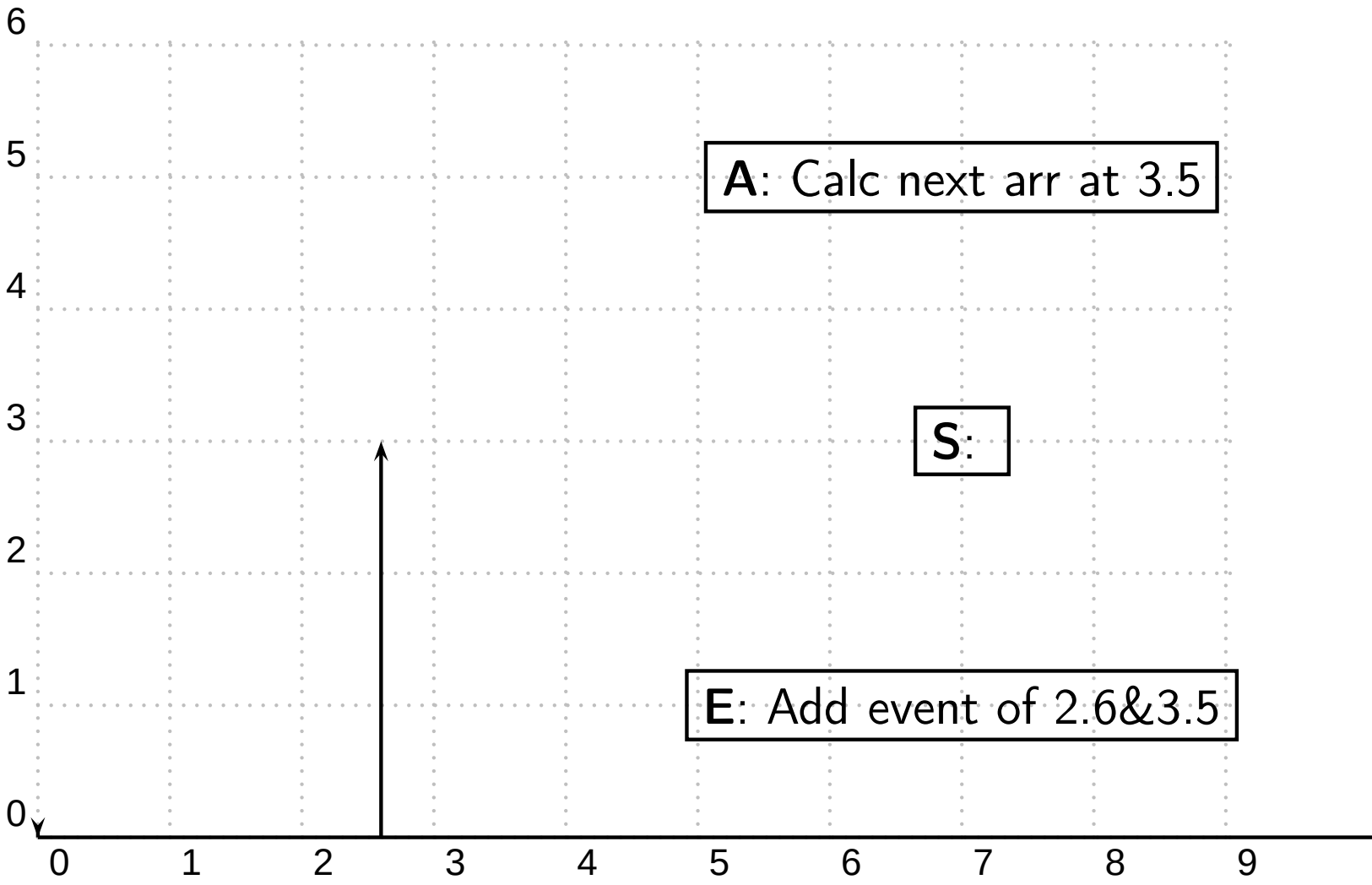
Process-Oriented Discrete-Event Simulation



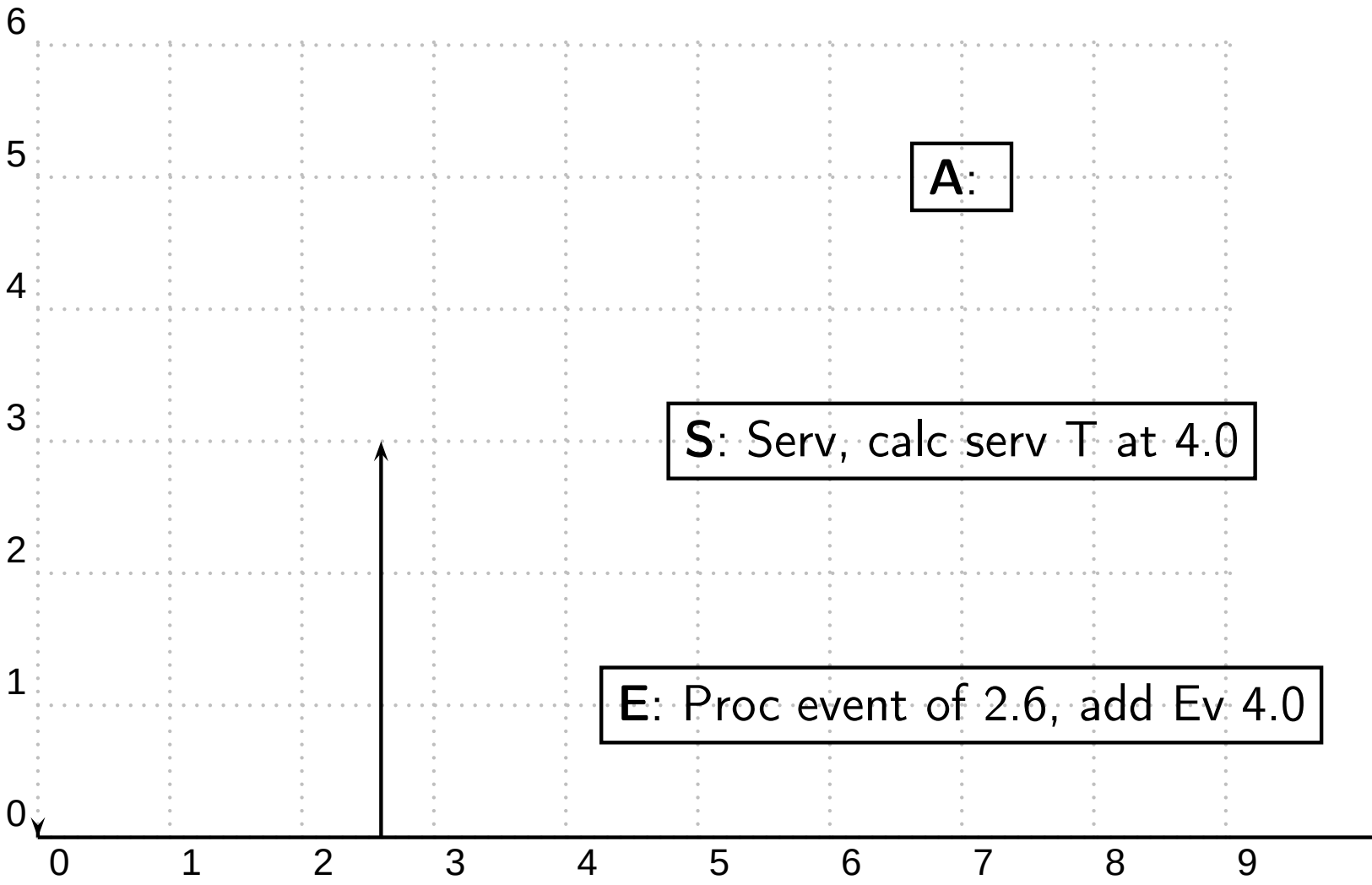
Process-Oriented Discrete-Event Simulation



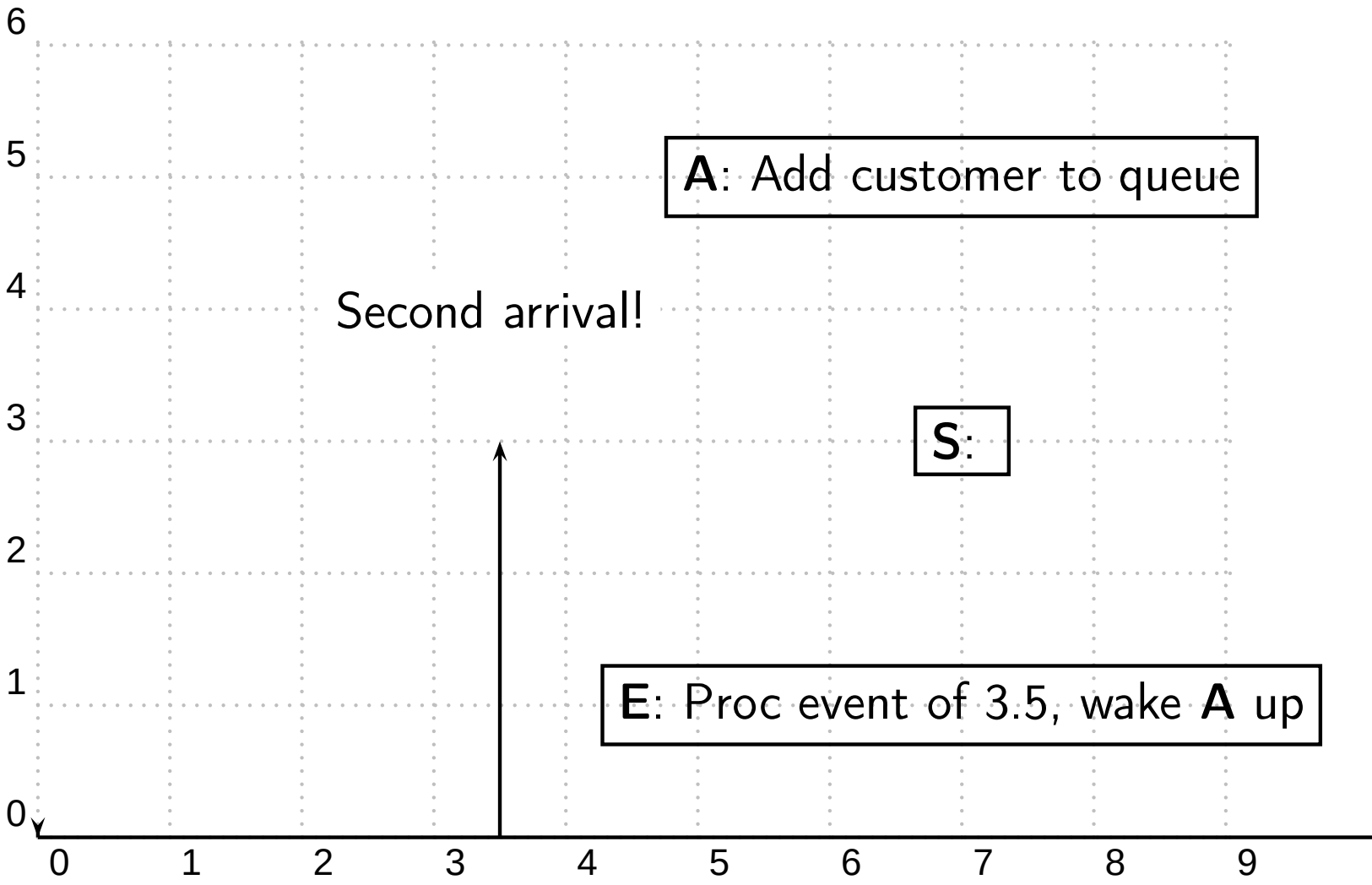
Process-Oriented Discrete-Event Simulation



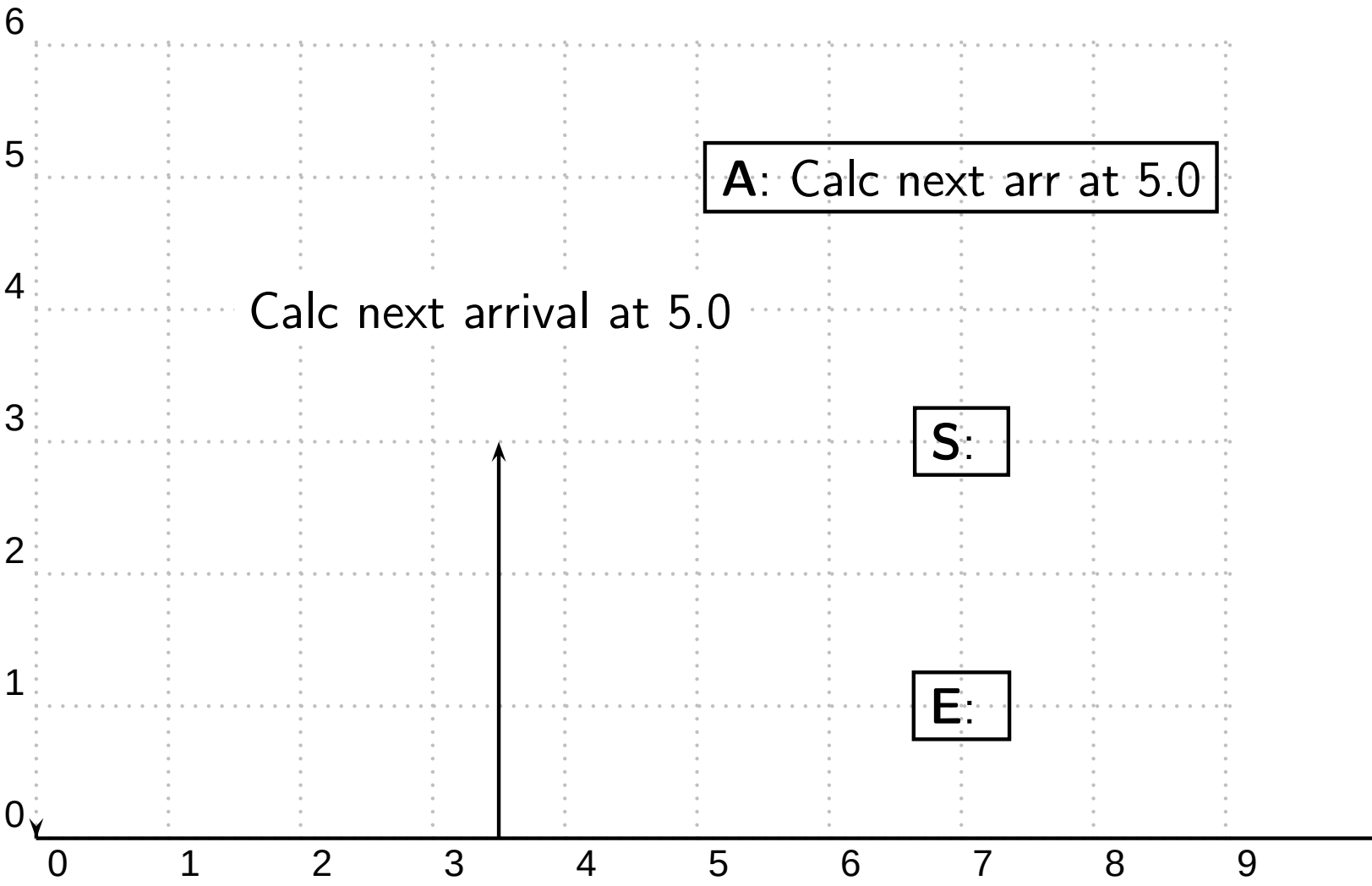
Process-Oriented Discrete-Event Simulation



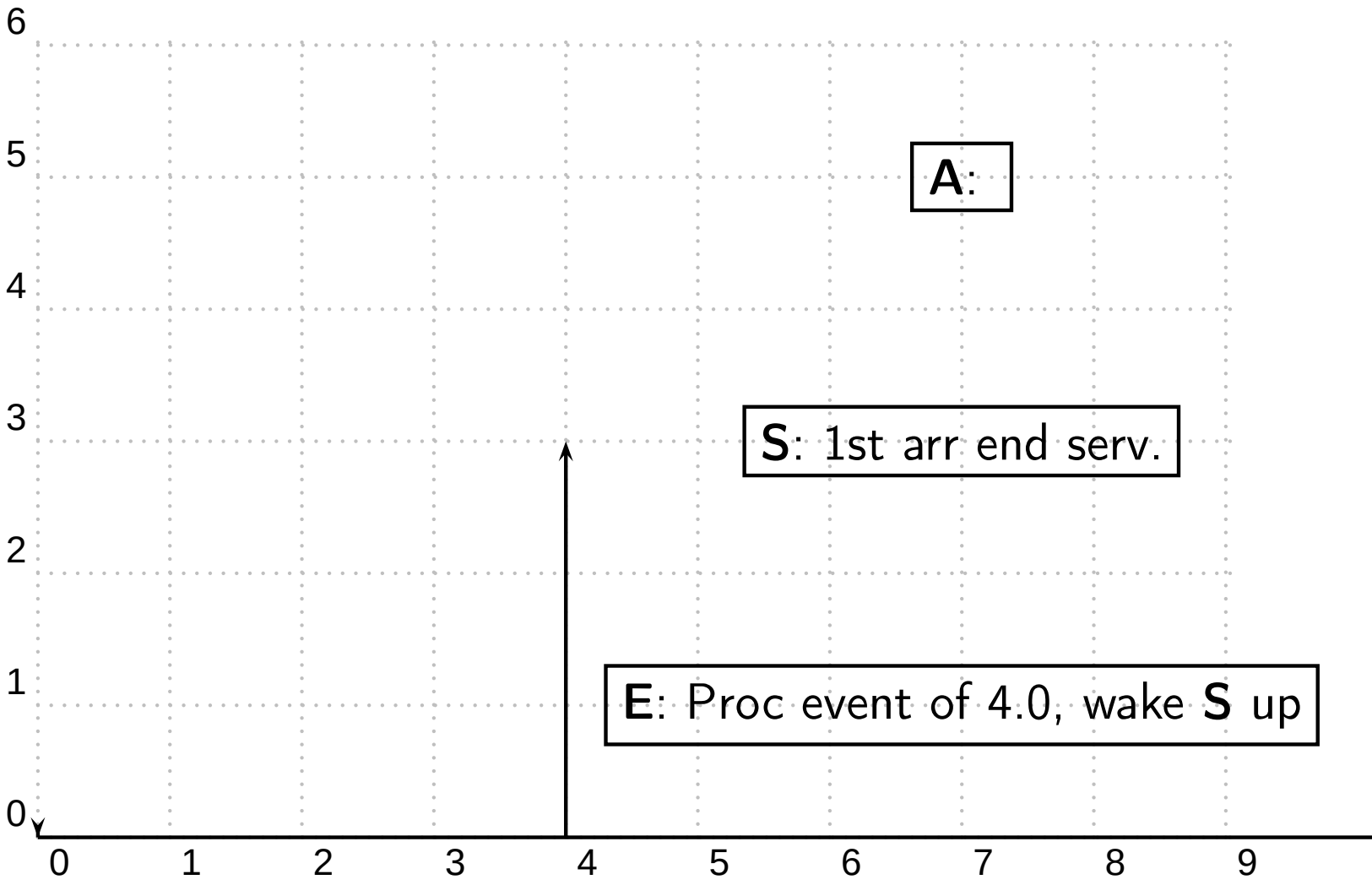
Process-Oriented Discrete-Event Simulation



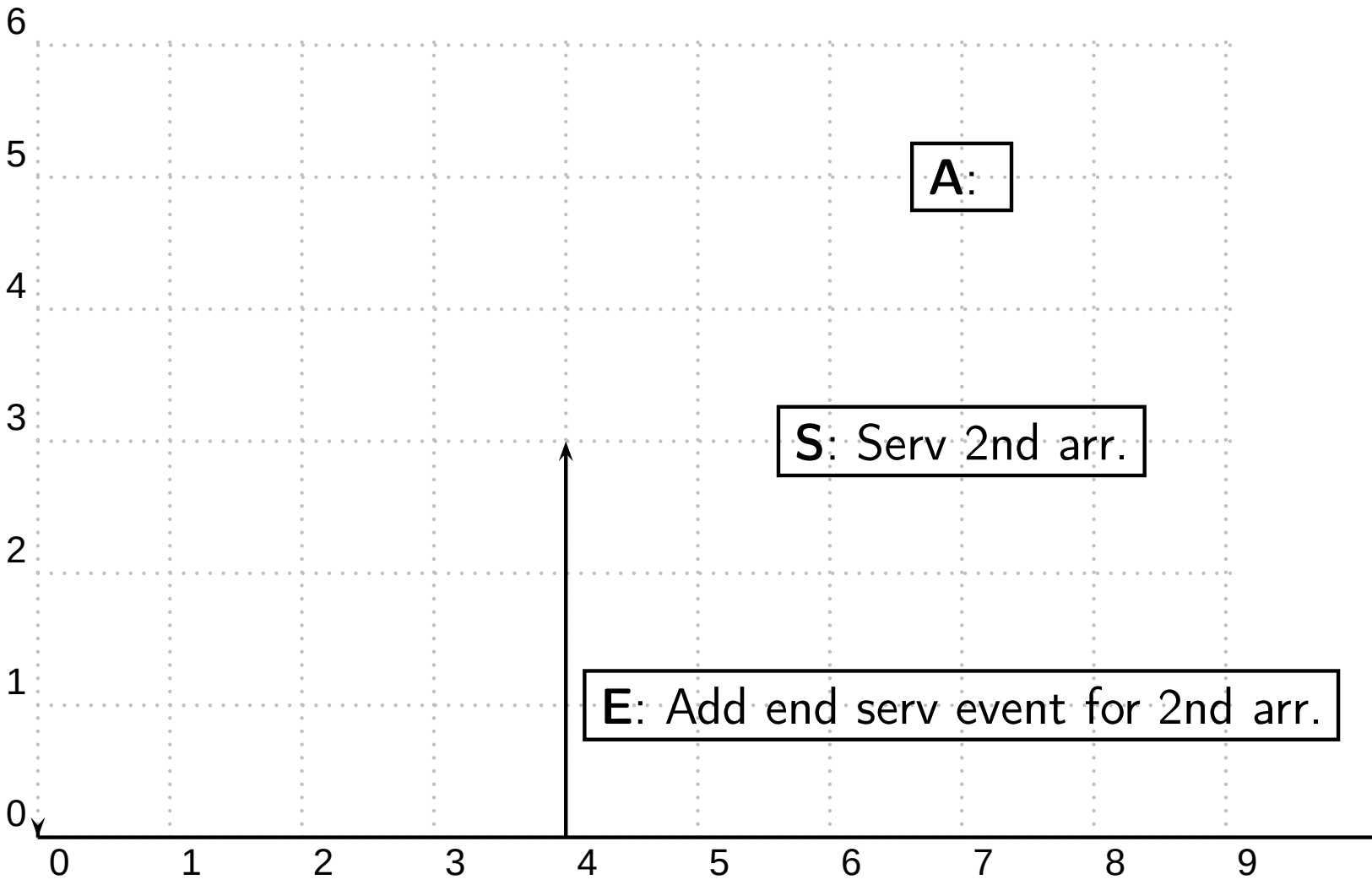
Process-Oriented Discrete-Event Simulation



Process-Oriented Discrete-Event Simulation



Process-Oriented Discrete-Event Simulation



Implementing Discrete-Event Simulation

- Use your own C/C++ implementation
  - takes time to write simulation engine **and** algorithm code
  - hard to debug, especially for event manager
  - not very convincing

Implementing Discrete-Event Simulation

- Use your own C/C++ implementation
  - takes time to write simulation engine **and** algorithm code
  - hard to debug, especially for event manager
  - not very convincing
- Use generalized simulation library, or language
  - SIMULA programming language
  - C++SIM or JavaSIM [1]
  - SimEvents in Simulink/MATLAB
  - SimPy [5]

Implementing Discrete-Event Simulation

- Use your own C/C++ implementation
  - takes time to write simulation engine **and** algorithm code
  - hard to debug, especially for event manager
  - not very convincing
- Use generalized simulation library, or language
  - SIMULA programming language
  - C++SIM or JavaSIM [1]
  - SimEvents in Simulink/MATLAB
  - SimPy [5]
- Use special purpose simulation packages
  - ns-3 for network simulation [4]

Introduction to SimPy

- Uses Python for modeling
  - Python is a scripting language like MATLAB, but **faster**!
  - Python is very easy to write and very beautiful!

Introduction to SimPy

- Uses Python for modeling
  - Python is a scripting language like MATLAB, but **faster**!
  - Python is very easy to write and very beautiful!
- Process-oriented Discrete Event Simulation Language
  - easier to write model
  - provides event manager implementation

Introduction to SimPy

- Uses Python for modeling
  - Python is a scripting language like MATLAB, but **faster**!
  - Python is very easy to write and very beautiful!
- Process-oriented Discrete Event Simulation Language
  - easier to write model
  - provides event manager implementation
- Uses coroutine to suspend/resume process
  - will be refered to as **thread** in this presentation
  - guarantees order of execution
  - cannot run on parallel machine

SimPy Terminology: Classes

- Process
  - simulates an entity which evolves in time, e.g. a customer who needs to be served by a clerk
  - refered to as **thread** in [3]

SimPy Terminology: Classes

- Process
  - simulates an entity which evolves in time, e.g. a customer who needs to be served by a clerk
  - refered to as **thread** in [3]
- Resource
  - simulates something to be queued, e.g. the waiting list

SimPy Terimnology: Functions

activate() used to mark a thread as runnable when it is first created

SimPy Terimnology: Functions

activate()	used to mark a thread as runnable when it is first created
simulate()	starts the simulation

SimPy Terimnology: Functions

activate()	used to mark a thread as runnable when it is first created
simulate()	starts the simulation
yield hold	put current thread into suspension for a certain amount of time

SimPy Terimnology: Functions

activate()	used to mark a thread as runnable when it is first created
simulate()	starts the simulation
yield hold	put current thread into suspension for a certain amount of time
yield request	requests for a given resource

SimPy Terimnology: Functions

activate()	used to mark a thread as runnable when it is first created
simulate()	starts the simulation
yield hold	put current thread into suspension for a certain amount of time
yield request	requests for a given resource
yield release	used to indicate that current thread no longer need the given resource

SimPy Terimnology: Functions

activate()	used to mark a thread as runnable when it is first created
simulate()	starts the simulation
yield hold	put current thread into suspension for a certain amount of time
yield request	requests for a given resource
yield release	used to indicate that current thread no longer need the given resource
yield passivate	put current thread into suspension and wait until awakened by some other thread

SimPy Terimnology: Functions

activate()	used to mark a thread as runnable when it is first created
simulate()	starts the simulation
yield hold	put current thread into suspension for a certain amount of time
yield request	requests for a given resource
yield release	used to indicate that current thread no longer need the given resource
yield passivate	put current thread into suspension and wait until awakened by some other thread
reactivate()	awakes a previously-passivated thread

SimPy Terimnology: Functions

activate()	used to mark a thread as runnable when it is first created
simulate()	starts the simulation
yield hold	put current thread into suspension for a certain amount of time
yield request	requests for a given resource
yield release	used to indicate that current thread no longer need the given resource
yield passivate	put current thread into suspension and wait until awakened by some other thread
reactivate()	awakes a previously-passivated thread
cancel()	cancels all the events associated with a previously-passivated thread

SimPy Example

- Scenario
  - A post office with only one clerk.
  - Customers arrival is poisson process,
    i.e. inter-arrival time is exponential distribution.
  - Service time is also poisson process.
- Process
  - Arrival
  - Clerk
- Queue is managed by ourselves.

Arrival Process


```
class ArrivalClass(Process):
    ArrivalRate = 1/1.0                    # reciprocal of mean arrival tim

    def __init__(self):
        Process.__init__(self)

    def Run(self):
        while 1:
            InterArrivalTime = G.Rnd.expovariate(ArrivalClass.ArrivalRat
            yield hold,self,InterArrivalTime
            C = Customer()
            ClerkClass.Queue.append(C) # a customer arrives
            G.NumCustomers += 1

            if ClerkClass.Idle != []:   # Is there any clerk idle?
                reactivate(ClerkClass.Idle[0]) # Yes, wake him/her up
```

Clerk Process


```
class ClerkClass(Process):
    ServiceRate = 1/1.2                    # reciprocal of mean service tim
    MaxQueueLength = 0
    Queue = []
    Idle = []
    Busy = []
    NumDone = 0
    def __init__(self):
        Process.__init__(self)
        ClerkClass.Idle.append(self)       # Initially idle
    def Run(self):
        while 1:
            yield passivate,self           # wait until awaken by customers
            ClerkClass.Idle.remove(self)
            ClerkClass.Busy.append(self) # going to be busy
            while ClerkClass.Queue != []:
                C = ClerkClass.Queue.pop() # call next customer in line
                if len(ClerkClass.Queue) > ClerkClass.MaxQueueLength:
                    ClerkClass.MaxQueueLength = len(ClerkClass.Queue)
                # Start service the customer
                ServiceTime = G.Rnd.expovariate(ClerkClass.ServiceRate)
                yield hold,self,ServiceTime
                C.endService()
                G.TotalWaitingTime += C.WaitingTime
                ClerkClass.NumDone += 1
                del C

            ClerkClass.Busy.remove(self)
            ClerkClass.Idle.append(self)
```

Live Demo

Important Simulation Parameters

Purpose List below parameters to prove that your work is repeatable.

RNG Random number generator method

- Linear Congruential Method
  - oldest and best well known
- Mersenne Twister
  - designed with simulation purpose in mind
  - used to implement random library in python after version 2.5

RNG Random number generator seed

Concluding Remarks

- Simulation is a powerful tool to study physical problems with cheaper cost.
- SimPy provides process-oriendted DES framework to write simlation easily and reasonably fast.

References

[1] JavaSIM and C++SIM. http://javasim.codehaus.org/.

[2] Jerry Banks, John S. Carson, Barry L. Nelson, and David M. Nicol. *Discrete-Event System Simulation (5th Edition)*. Prentice Hall, 2009.

[3] N Matloff. A discrete-event simulation course based on the simpy language. *Davis*, 2006.

[4] ns -3 Network Simulator. http://www.nsnam.org/.

[5] SimPy Simulation Package. http://simpy.sourceforge.net/.

Q & A