Performance Modeling of Computer Systems and Networks

Project 2018-2019

Andrea Graziani - 0273395

Universit‘a degli Studi di Roma “Tor Vergata”
FACOLTA’ DI INGEGNERIA
Corso di Laurea Magistrale in Ingegneria Informatica

9 luglio 2019

Specification model

Introduction

- What are the **state variables**?

$n_x^{(c)}(\tau)$	$=$	Number of class c jobs currently running at x system's node at time τ
$d_x^{(c)}(\tau)$	$=$	Number of class c departed jobs from node x at time τ
$s_{x,i}^{(c)}$	$=$	Service time of class c job i served on x node
$i_{cloudlet}^{(2)}(\tau)$	$=$	Number of class 2 interrupted jobs at time τ which were running on cloudlet (Access control algorithm 2 only!)

Where:

$$c \in \{1, 2\} = C$$
$$x \in \{cloudlet, cloud, global\} = X$$
$$\tau \in (t_0, t)$$

Specification model

Introduction

- Are there any **constrains** regarding the values assumed by these variables?

$$\sum_{c \in C} n_{cloudlet}^{(c)}(\tau) \leq N \quad \forall \tau \in (t_0, t) \tag{1}$$

  - If you are using *Access Control Algorithm 2*:

$$\begin{aligned} n_{cloudlet}^{(2)}(\tau) > 0 \quad &\Rightarrow \quad \sum_{c \in C} n_{cloudlet}^{(c)}(\tau) \leq S \quad \forall \tau \in (t_0, t) \\ S + 1 \leq n_{cloudlet}^{(1)}(\tau) \leq N \quad &\Leftrightarrow \quad n_{cloudlet}^{(2)}(\tau) = 0 \quad \forall \tau \in (t_0, t) \end{aligned} \tag{2}$$

Specification Model

System State

- What's meant by **system state**?
- So, how we can define system state formally?

$$\omega(\tau) = (\omega_{cloudlet}(\tau), \omega_{cloud}(\tau)) \tag{3}$$

Where:

$$\begin{aligned} \omega_{cloudlet}(\tau) &= (n^{(1)}_{cloudlet}(\tau), n^{(2)}_{cloudlet}(\tau)) \\ \omega_{cloud}(\tau) &= (n^{(1)}_{cloud}(\tau), n^{(2)}_{cloud}(\tau)) \end{aligned} \tag{4}$$

Thus:

$$\omega(\tau) = ((n^{(1)}_{cloudlet}(\tau), n^{(2)}_{cloudlet}(\tau)), (n^{(1)}_{cloud}(\tau), n^{(2)}_{cloud}(\tau)) \tag{5}$$

- How do they evolve in time?

Specification Model

Events

- What is an event?
- What is the first event that can occur? And the last one?

$$
\begin{aligned}
& n_x^{(c)}(t_0) = 0 \qquad \forall c \in C, \forall x \in X \\
& d_x^{(c)}(t_0) = 0 \qquad \forall c \in C, \forall x \in X \\
& n_x^{(c)}(t) = 0 \qquad \forall c \in C, \forall x \in X \\
& t_0 = 0, t = \tau^*
\end{aligned}
\tag{6}
$$

- So, what are the events capable to change system status?

Specification Model

Events

Event name that occurred at time τ	Event's place	Event's effects
Class 1 job arrival	Cloudlet	$n^{(1)}_{\text{cloudlet}}(\tau)$++
	Cloud	$n^{(1)}_{\text{cloud}}(\tau)$++
	Controller	$n^{(1)}_{\text{global}}(\tau)$++
Class 2 job arrival	Cloudlet	$n^{(2)}_{\text{cloudlet}}(\tau)$++
	Cloud	$n^{(2)}_{\text{cloud}}(\tau)$++
	Controller	$n^{(2)}_{\text{global}}(\tau)$++
Class 1 job departure	Cloudlet	$n^{(1)}_{\text{cloudlet}}(\tau)$−− $n^{(1)}_{\text{global}}(\tau)$−− $d^{(1)}_{\text{cloudlet}}(\tau)$++ $d^{(1)}_{\text{global}}(\tau)$++
	Cloud	$n^{(1)}_{\text{cloud}}(\tau)$−− $n^{(1)}_{\text{global}}(\tau)$−− $d^{(1)}_{\text{cloud}}(\tau)$++ $d^{(1)}_{\text{global}}(\tau)$++
Class 2 job departure	Cloudlet	$n^{(2)}_{\text{cloudlet}}(\tau)$−− $n^{(2)}_{\text{global}}(\tau)$−− $d^{(2)}_{\text{cloudlet}}(\tau)$++ $d^{(2)}_{\text{global}}(\tau)$++
	Cloud	$n^{(2)}_{\text{cloud}}(\tau)$−− $n^{(2)}_{\text{global}}(\tau)$−− $d^{(2)}_{\text{cloud}}(\tau)$++ $d^{(2)}_{\text{global}}(\tau)$++

Specification Model

Events

- When you are using *Access Control Algorithm 2*:
  - If **class 1 job arrival event on controller** occurs:

$$n^{(1)}_{\text{global}}(\tau)++ \tag{7}$$

  - **If** following conditions are **all** true:

$$\begin{aligned} n^{(1)}_{cloudlet}(\tau) &\neq N \\ n^{(1)}_{cloudlet}(\tau) + n^{(2)}_{cloudlet}(\tau) &= S \\ n^{(2)}_{cloudlet} &> 0 \end{aligned} \tag{8}$$

Then:

- $n^{(2)}_{cloudlet}(\tau) - -$
- Class 1 job arrival on cloudlet node event is scheduled.
- Class 2 job arrival on cloud node event is scheduled.
- A Class 2 job departure on cloudlet node event is removed from the *event list*.
- $i^{(2)}_{cloudlet}(\tau) + +$

Specification Model

Statistics computation

- How to compute requested statistics?

$$E[N_x]^{(c)} = \frac{1}{t - t_0} \int_{t_0}^{t} n_x^{(c)}(\tau) d\tau \qquad \forall c \in C, \forall x \in X \tag{9}$$

$$\begin{aligned} E[N_x] &= \frac{1}{t - t_0} \cdot \int_{t_0}^{t} \left(n_x^{(1)}(\tau) + n_x^{(2)}(\tau) \right) d\tau \\ &= \frac{1}{t - t_0} \cdot \int_{t_0}^{t} n_x^{(1)}(\tau) d\tau + \frac{1}{t - t_0} \cdot \int_{t_0}^{t} n_x^{(2)}(\tau) d\tau \\ &= E[N_x]^{(1)} + E[N_x]^{(2)} \\ &= \sum_{c \in C} E[N_x]^{(c)} \qquad \forall x \in X \end{aligned} \tag{10}$$

Specification Model

Statistics computation

$$E[T_x]^{(c)} = E[S_x]^{(c)} = \frac{1}{d_x^{(c)}(t)} \cdot \sum_{i=0}^{d_x^{(c)}(t)} s_{x,i}^{(c)} \quad \forall c \in C, \forall x \in X \tag{11}$$

If we aren't interested about per-class metrics, we can compute $E[S_x]$ and $E[T_x]$, respectively *time-averaged service time* and the *time-averaged response time experienced by any class jobs in a x system's node*, as follows:

$$E[T_x] = E[S_x] = \frac{1}{d_x^{(1)}(t) + d_x^{(2)}(t)} \cdot \left(\sum_{i=0}^{d_x^{(1)}(t)} s_{x,i}^{(1)} + \sum_{i=0}^{d_x^{(2)}(t)} s_{x,i}^{(2)} \right) \quad \forall x \in X \tag{12}$$

Computational Model

Next-event simulation logic

- Which are main features of a discrete-event simulation using a **next-event** approach?
- How we have implemented *simulation clock* and the *event list*?
- How we have implemented next-event simulation algorithm in our computational model?

Computational Model

Next-event simulation logic

Listing 1: `ComputationalModel.perform` method

```
1   public void perform() {
2
3      initializeSimulation();
4
5      while (!this.simulationEventList.isEmpty()) {
6
7         SimulationEvent actualEvent = this.simulationEventList.poll();
8
9         if (actualEvent != null) {
10
11             SimulationClock.getInstance().setCurrentEventTime(actualEvent.getStartTime());
12
13             actualEvent.perform();
14             actualEvent.scheduleFollowingEvent();
15
16             SimulationEvent nextEvent = this.simulationEventList.peek();
17
18             if (nextEvent != null)
19                 SimulationClock.getInstance().setNextEventTime(nextEvent.getStartTime());
20        }
21        updateStatistics();
22     }
23  ...
```

Listing 2: `ComputationalModel.initializeSimulation` method

```
1   private void initializeSimulation() {
2        initializeSystemStateVariables();
3        initializeSimulationClock();
4        scheduleInitialEvent();
5   }
```

Computational Model

Next-event simulation logic

Listing 3: `System` class methods

```
1   @Override
2   protected void initializeSystemStateVariables() {
3       this.cloud.initializeSystemStateVariables();
4       this.cloudlet.initializeSystemStateVariables();
5       this.globalNetwork.initializeSystemStateVariables();
6   }
7
8   @Override
9   protected void initializeSimulationClock() {
10      SimulationClock.getInstance().setCurrentEventTime(0.0);
11      SimulationClock.getInstance().setNextEventTime(0.0);
12  }
13
14  @Override
15  protected void scheduleInitialEvent() {
16      this.globalNetwork.scheduleInitialEvent();
17  }
```

Listing 4: `GlobalNetwork.scheduleInitialEvent` method

```
1   @Override
2   public void scheduleInitialEvent() {
3       SystemEvent firstEventClass1 = SystemEventFactory.buildClass1JobArrival();
4       SystemEvent firstEventClass2 = SystemEventFactory.buildClass2JobArrival();
5
6       this.system.scheduleEventOnGlobalNetwork(firstEventClass1, this.
              getNextClass1JobInterArrivalTime());
7       this.system.scheduleEventOnGlobalNetwork(firstEventClass2, this.
              getNextClass2JobInterArrivalTime());
8   }
```

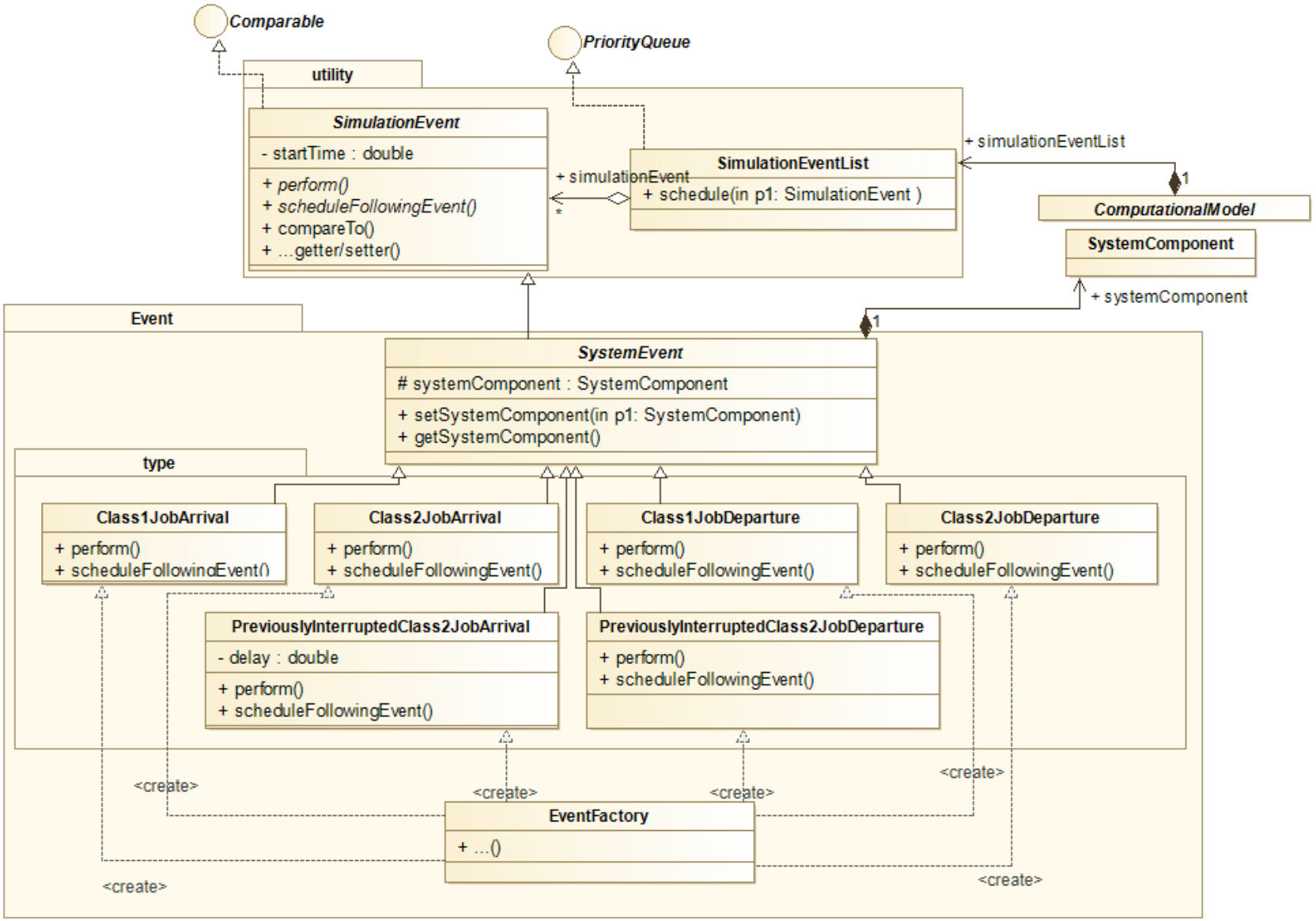
Computational Model

Events

- How we have implemented events in our computational model?

Computational Model

Events



Computational Model

Events

Listing 5: `SystemEvent` class

```
1  public abstract class SystemEvent extends SimulationEvent {
2
3      protected SystemComponent systemComponent;
4
5      public void setSystemComponent(SystemComponent systemComponent) {
6          this.systemComponent = systemComponent;
7      }
8
9      public SystemComponent getSystemComponent() {
10         return systemComponent;
11     }
12     ...
13 }
```

Listing 6: `Class1JobArrival` class

```
1  public class Class1JobArrival extends SystemEvent {
2
3      @Override
4      public void perform() {
5          this.systemComponent.updateStatusAfterClass1JobArrival();
6      }
7
8      @Override
9      public void scheduleFollowingEvent() {
10         this.systemComponent.scheduleFollowingEventAfterClass1JobArrival();
11     }
12 }
```

Computational Model

Events

- `PreviouslyInterruptedClass2JobArrival`? What is it for?

Listing 7: `PreviouslyInterruptedClass2JobArrival` class

```
1  public class PreviouslyInterruptedClass2JobArrival extends SystemEvent {
2
3      private double delay;
4
5      public PreviouslyInterruptedClass2JobArrival(double delay) {
6          this.delay = delay;
7      }
8
9      @Override
10     public void perform() {
11         this.systemComponent.updateStatusAfterPreviouslyInterruptedClass2JobArrival(this.delay);
12     }
13
14     @Override
15     public void scheduleFollowingEvent() {
16         this.systemComponent.scheduleFollowingEventAfterPreviouslyInterruptedClass2JobArrival();
17     }
18 }
```

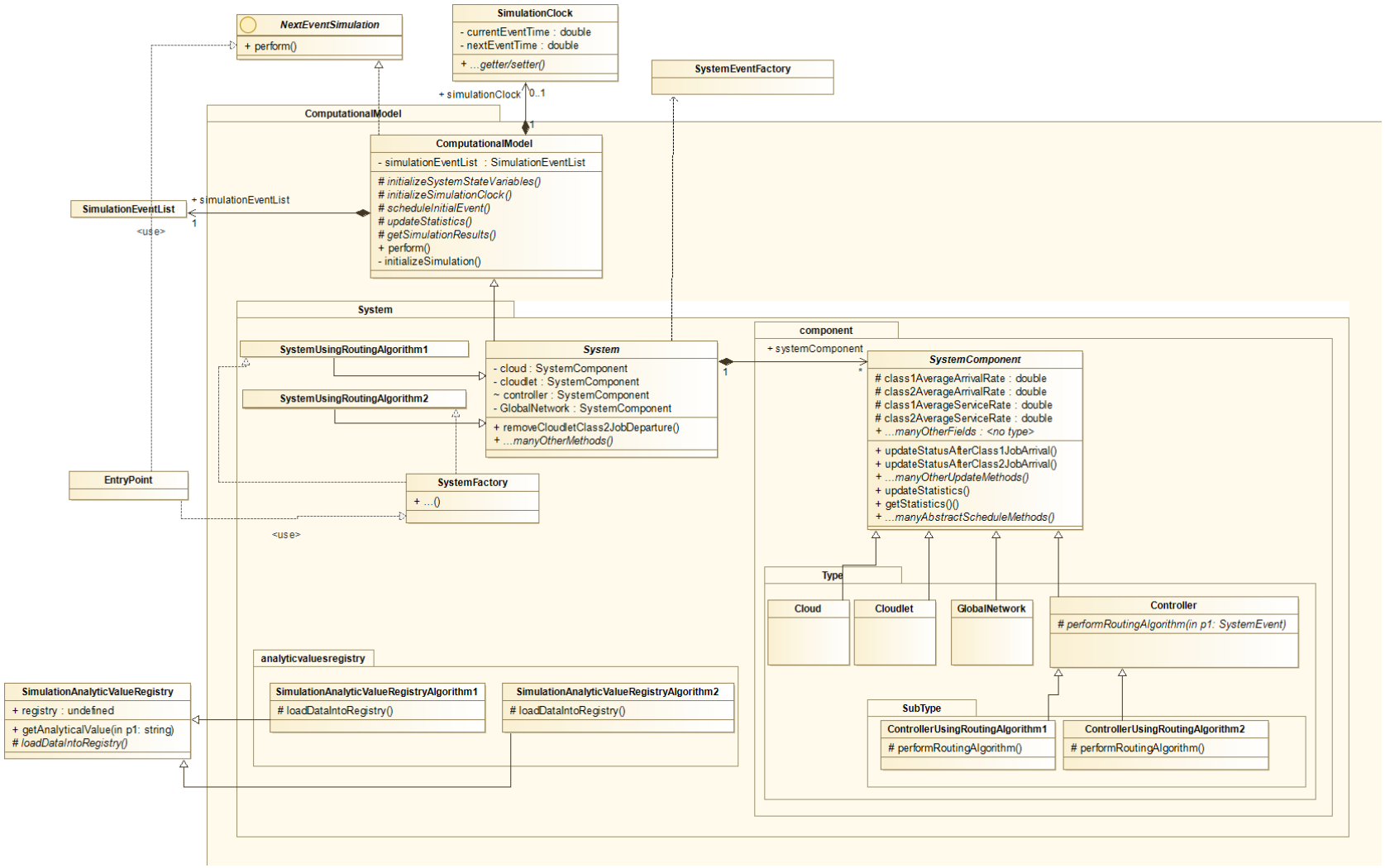
Computational Model

System implementation

- How we have implemented our system?
- Which classes hold system state variables? Which compute output statistics?
- How can we start our simulator?

Computational Model

System implementation



Computational Model

Class 2 job interruption

- How we manage class 2 job interruption?

Routing algorithm 2

Class 2 job interruption

```
1   protected void performRoutingAlgorithm(SystemEvent event) {
2
3       int n1 = this.system.getNumberOfClass1JobOnCloudlet();
4       int n2 = this.system.getNumberOfClass2JobOnCloudlet();
5
6           if (event instanceof Class1JobArrival) {
7
8               if (n1 == this.system.getThreshold())
9                   this.system.scheduleEventOnCloud(event, 0);
10              else if (n1 + n2 < this.system.getThreshold())
11                  this.system.scheduleEventOnCloudlet(event, 0);
12              else if (n2 > 0) {
13
14                  double runningCloudletTimeOfInterruptedJob = this.system.
                        removeCloudletClass2JobDeparture();
15
16                  this.system.scheduleEventOnCloudlet(event, 0);
17
18                  double setupTime = RandomNumberGenerator.getInstance().getExponential(5, 0.8);
19
20                  this.system.scheduleEventOnCloud(SystemEventFactory.
                        buildPreviouslyInterruptedClass2JobArrival(setupTime +
                        runningCloudletTimeOfInterruptedJob), setupTime);
21
22                  this.numberOfInterruptedClass2Jobs++;
23
24              } else
25                  this.system.scheduleEventOnCloudlet(event, 0);
26
27
28          } else {
29
30              ...
31          }
32      }
```

Statistics Results

A Finite-Horizon simulation

- What kind of simulation have we performed?
- What is generated at the end of each simulation? How we can provide an estimation for required metrics?
  Let i i-th replication, where $i = 0, ..., n$

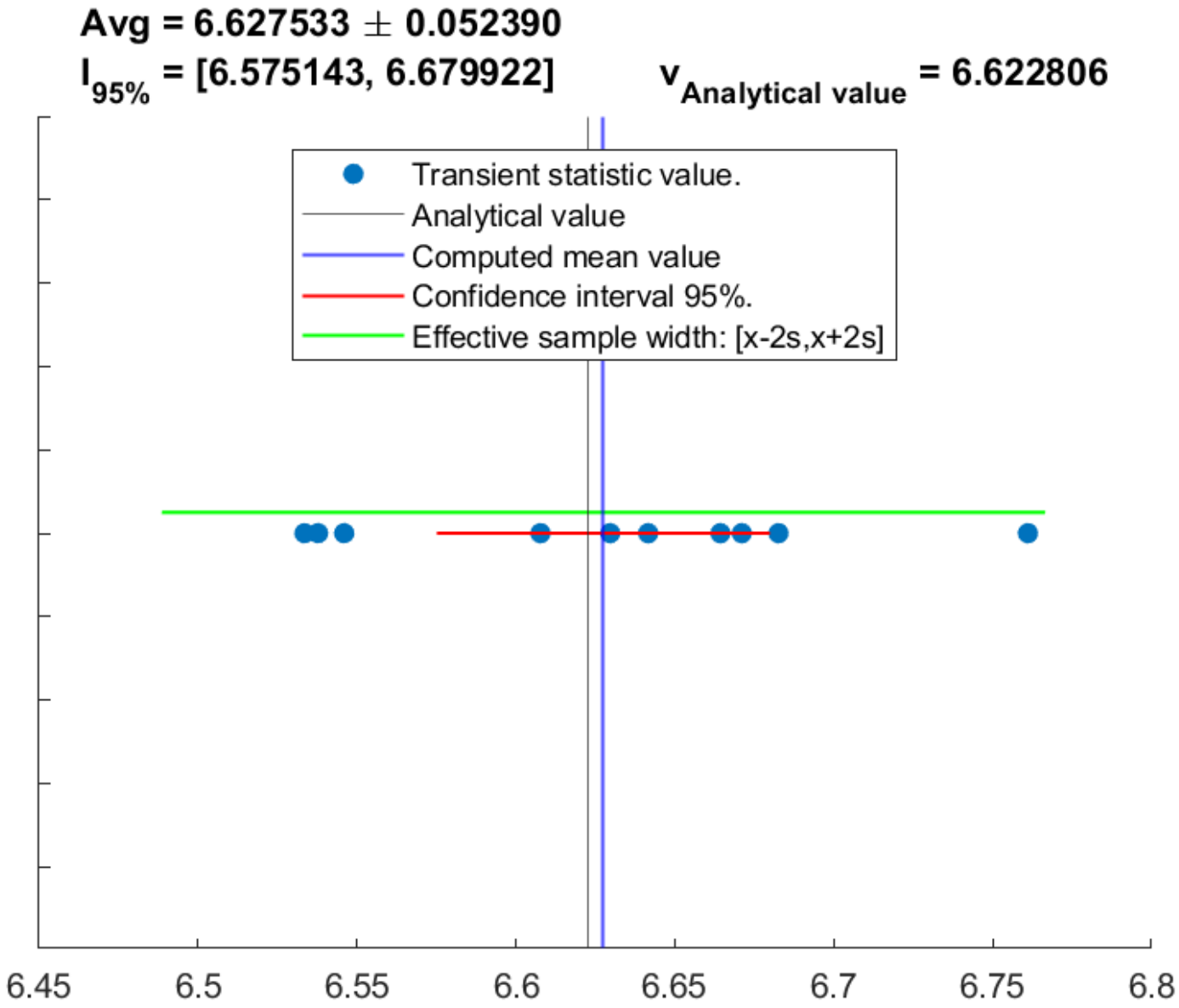
$$\bar{x}_i(t) = \int_0^t x_i(t)dt \qquad \forall i = 1, ..., n \tag{13}$$

$$E[\bar{x}(t)] = \frac{1}{n}\sum_{i=1}^{n} \bar{x}_i(t) \tag{14}$$

Obliviously, interval estimate for $E[\bar{x}(t)]$ can be calculated from the ensemble mean and standard deviation.

Statistics Results

Il processo client



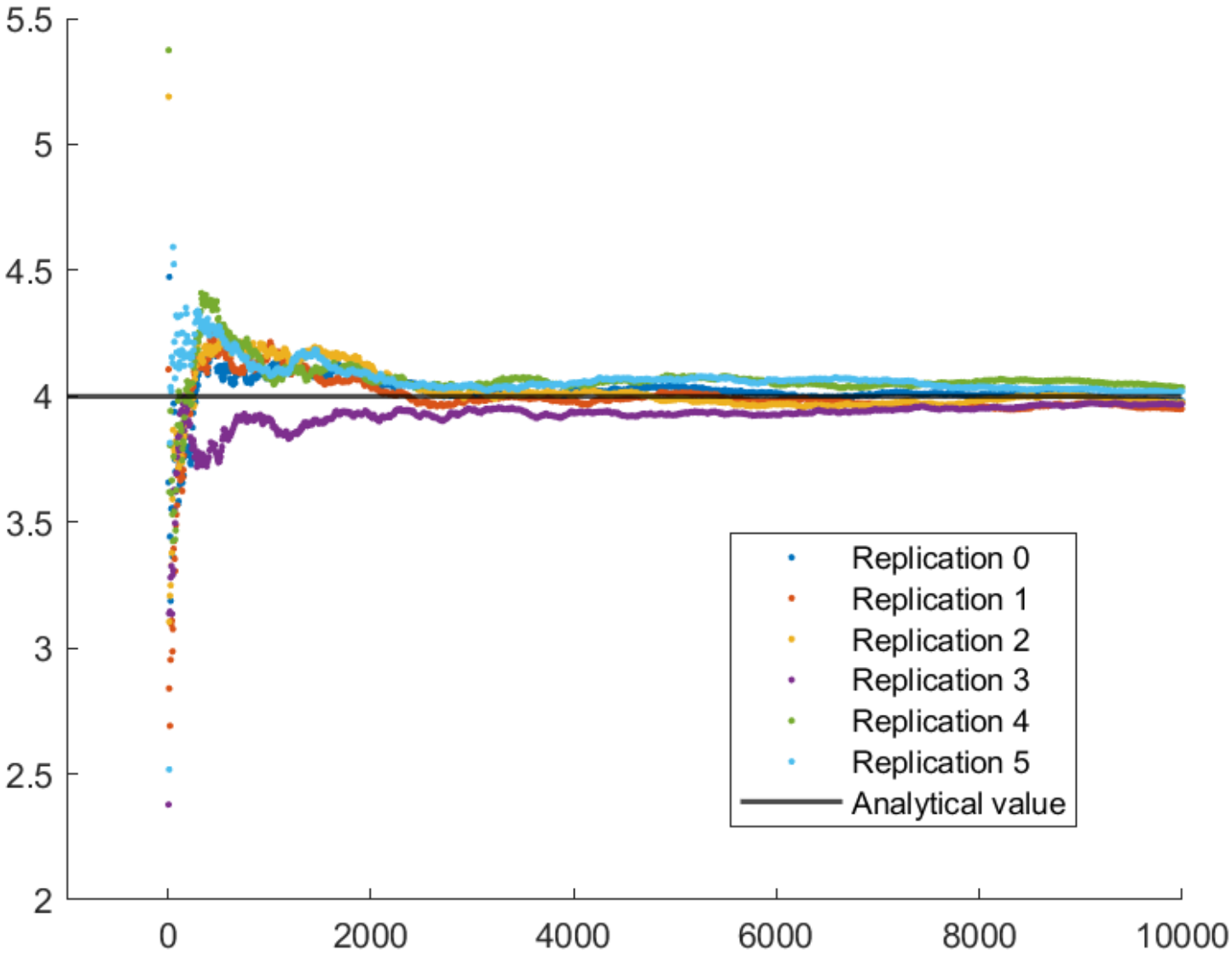
Statistics Results

A Finite-Horizon simulation

- What are steady-state statistics? When they exist?
- So, do the steady-state statistics exist in our system?
- Is possible to obtain an interval estimation for steady state statistics using our simulator?

Statistics Results

Steady State statistics



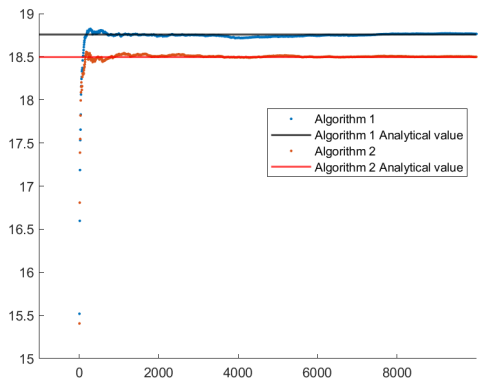
Statistics Results

Steady State statistics

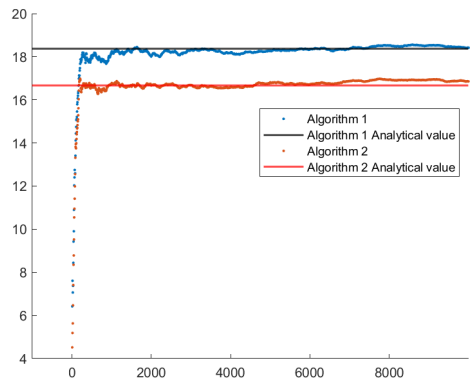
- Finite-horizon interval estimates are accurate steady-state estimates, becoming increasingly more accurate as the number of jobs increases.
- The steady-state average wait indicator is interior to the finite-horizon interval estimate, suggesting that current system simulation parameters (τ^*, n) allow us to achieve statistics that are close to their steady-state values.
- But there is another way...

Access control algorithms comparison

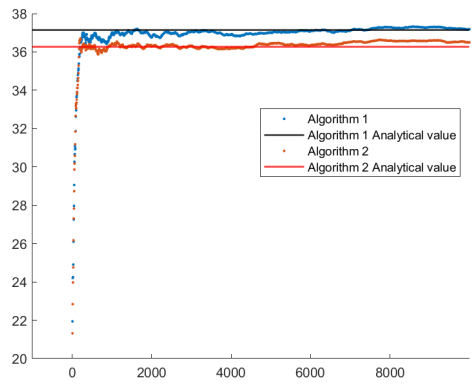
- Which routing policy is better?



(a) Cloudlet



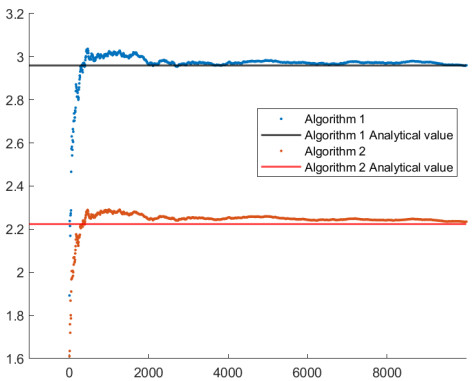
(b) Cloud



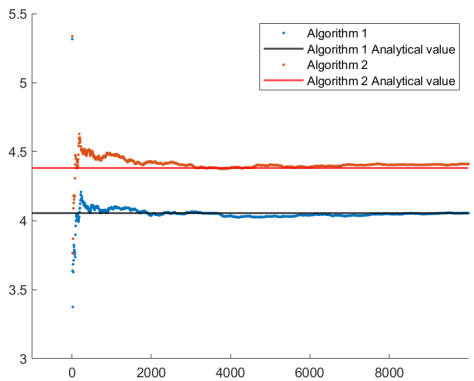
(c) Global

Figura: Time-Average Job Population

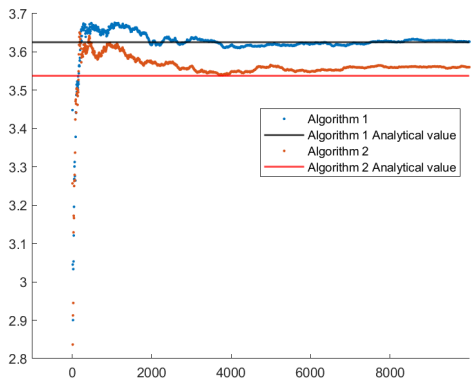
Access control algorithms comparison



(a) Class 1 Jobs



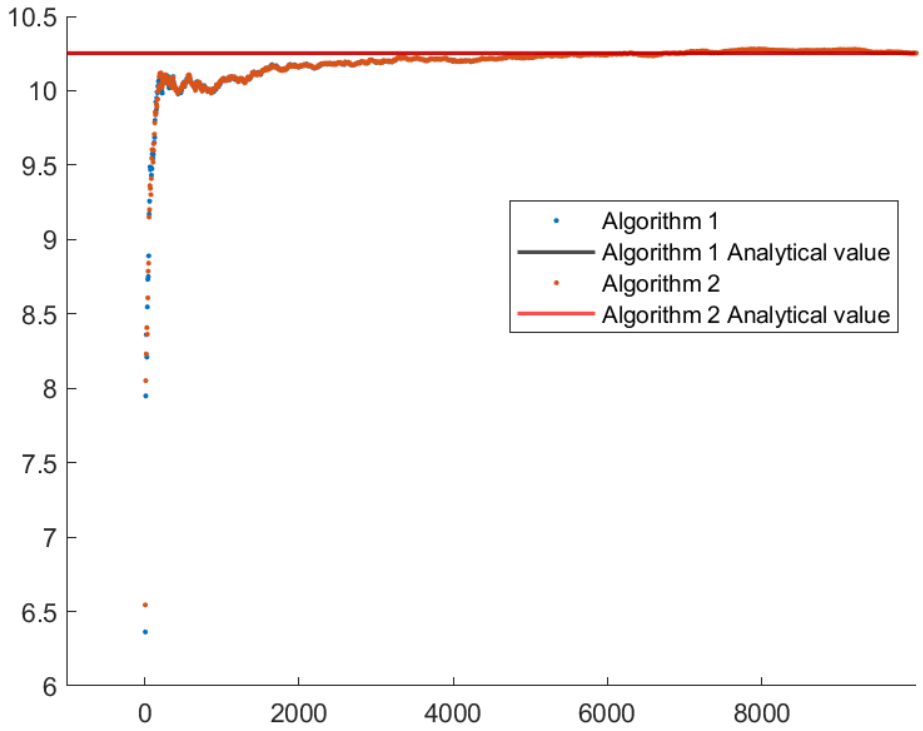
(b) Class 2 Jobs



(c) Both Classes

Figura: Global Time-Average service/response time

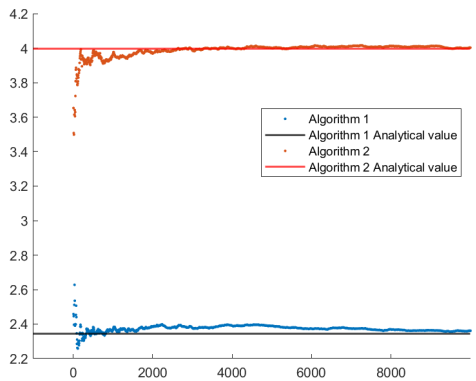
Access control algorithms comparison



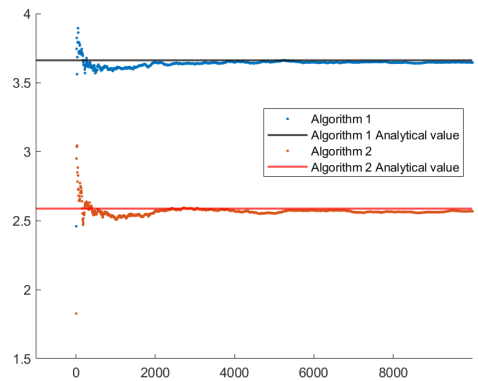
(a)

Figura: Global Throughput

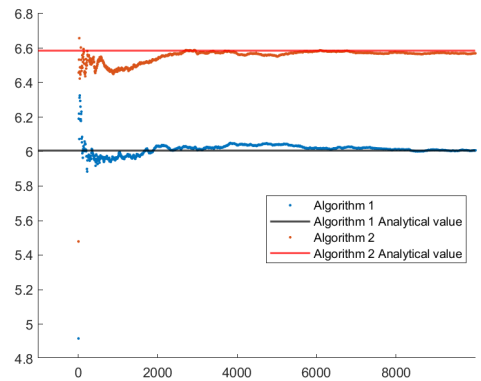
Access control algorithms comparison



(a) Class 1 Jobs



(b) Class 2 Jobs



(c) Both Classes

Figura: Cloudlet Throughput

Grazie per l'attenzione!