

Universit'a degli Studi di Roma "Tor Vergata"
FACOLTA' DI INGEGNERIA
Corso di Laurea Magistrale in Ingegneria Informatica

Performance Modeling of Computer Systems and Networks

Project 2018-2019

Andrea Graziani - 0273395

1 luglio 2019



- ▶ Mathematical notation and variables used in our specification model:

$$c \in \{1, 2\} = C$$

$$x \in \{\text{cloudlet}, \text{cloud}, \text{global}\} = X$$

$$\tau \in (t_0, t)$$

$$n_x^{(c)}(\tau)$$

$$d_x^{(c)}(\tau)$$

$$s_{x,i}^{(c)}$$

$$i_{\text{cloudlet}}^{(2)}(\tau)$$

- ▶ There are several constraints and relations among these variables.
 - ▶ Some of which depend on chosen access control algorithm.



$$\omega(\tau) = (\omega_{cloudlet}(\tau), \omega_{cloud}(\tau)) \quad (1)$$

Where:

$$\begin{aligned} \omega_{cloudlet}(\tau) &= (n_{cloudlet}^{(1)}(\tau), n_{cloudlet}^{(2)}(\tau)) \\ \omega_{cloud}(\tau) &= (n_{cloud}^{(1)}(\tau), n_{cloud}^{(2)}(\tau)) \end{aligned} \quad (2)$$

Thus:

$$\omega(\tau) = ((n_{cloudlet}^{(1)}(\tau), n_{cloudlet}^{(2)}(\tau)), (n_{cloud}^{(1)}(\tau), n_{cloud}^{(2)}(\tau))) \quad (3)$$



- ▶ $n_x^{(c)}(0) = 0$ and $d_x^{(c)}(0) = 0, \forall c \in \mathcal{C}, \forall x \in \mathcal{X}$
 - ▶ Initial system status is *idle*.
- ▶ First event must be either a class 1 or class 2 job arrival.
- ▶ Our stopping criteria: τ^* beyond which no new jobs can arrive.
 - ▶ Terminal state is idle.
 - ▶ System keep on working until all jobs have been completely served.



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

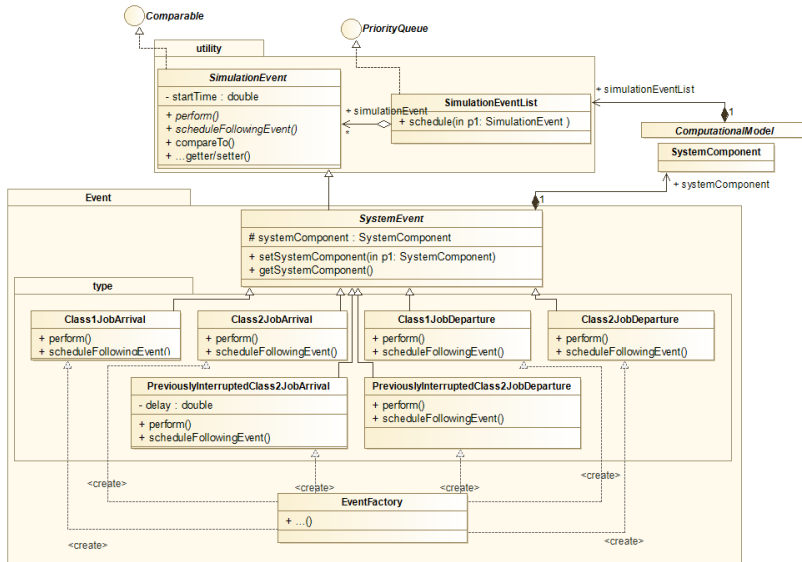


$$\begin{aligned}n_{cloudlet}^{(1)}(\tau) &\neq N \\n_{cloudlet}^{(1)}(\tau) + n_{cloudlet}^{(2)}(\tau) &> S \\n_{cloudlet}^{(2)} &\geq 0\end{aligned}\tag{4}$$

- ▶ $n_{cloudlet}^{(1)}(\tau)$ is incremented by one.
- ▶ $n_{cloudlet}^{(2)}(\tau)$ is decremented by one.
- ▶ class 2 job arrival on cloud node event is scheduled (considering a set-up time)

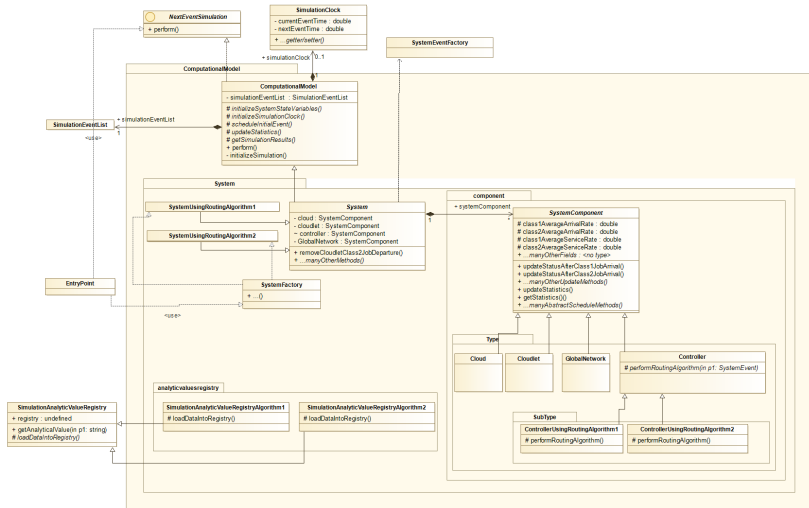
Class Diagrams 1

Computational Model



Class Diagrams 2

Computational Model





Listing 1: Snippet of perform method

```
public void perform() {  
    initializeSimulation();  
  
    while (!this.simulationEventList.isEmpty()) {  
        SimulationEvent actualEvent = this.simulationEventList.poll();  
  
        if (actualEvent != null) {  
            SimulationClock.getInstance().setCurrentEventTime(actualEvent.getStartTime());  
  
            actualEvent.perform();  
            actualEvent.scheduleFollowingEvent();  
  
            SimulationEvent nextEvent = this.simulationEventList.peek();  
  
            if (nextEvent != null)  
                SimulationClock.getInstance().setNextEventTime(nextEvent.getStartTime());  
        }  
        updateStatistics();  
    }  
    ...  
}
```

Routing algorithm 1

Computational Model



```
protected void performRoutingAlgorithm(SystemEvent event) {  
  
    int n1 = this.system.getNumberOfClass1JobOnCloudlet();  
    int n2 = this.system.getNumberOfClass2JobOnCloudlet();  
  
    if ((n1 + n2) == this.system.getThreshold())  
        this.system.scheduleEventOnCloud(event, 0);  
    else  
        this.system.scheduleEventOnCloudlet(event, 0);  
}
```

Routing algorithm 2

Computational Model



```
protected void performRoutingAlgorithm(SystemEvent event) {

    int n1 = this.system.getNumberOfClass1JobOnCloudlet();
    int n2 = this.system.getNumberOfClass2JobOnCloudlet();

    if (event instanceof Class1JobArrival) {

        if (n1 == this.system.getThreshold())
            this.system.scheduleEventOnCloud(event, 0);
        else if (n1 + n2 < this.system.getThreshold())
            this.system.scheduleEventOnCloudlet(event, 0);
        else if (n2 > 0) {

            double runningCloudletTimeOfInterruptedJob = this.system.
                removeCloudletClass2JobDeparture();

            this.system.scheduleEventOnCloudlet(event, 0);

            double setupTime = RandomNumberGenerator.getInstance().getExponential(5, 0.8);

            this.system.scheduleEventOnCloud(SystemEventFactory.
                buildPreviouslyInterruptedClass2JobArrival(setupTime +
                    runningCloudletTimeOfInterruptedJob), setupTime);

            this.numberofInterruptedClass2Jobs++;

        } else
            this.system.scheduleEventOnCloudlet(event, 0);

    } else {

        ...

    }
}
```



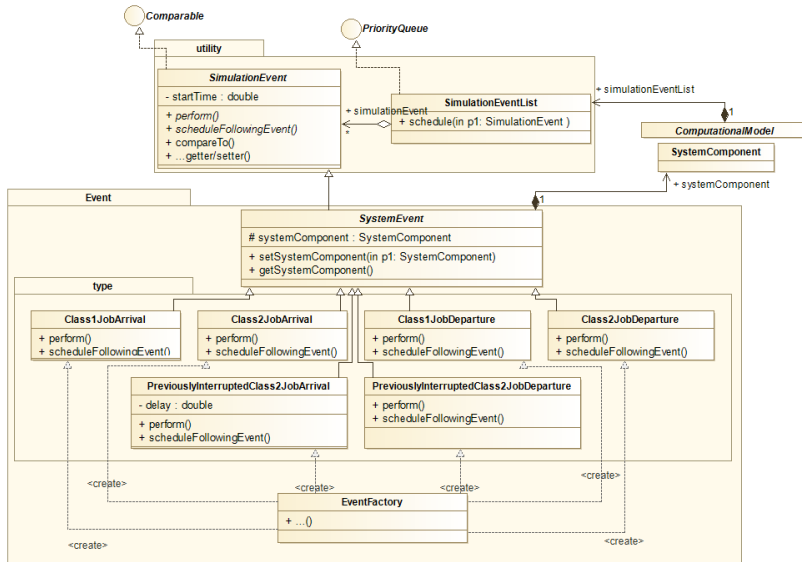
Do **steady-state** statistics exist?

Statistics Results

Il processo client



12





Steady-state system statistics

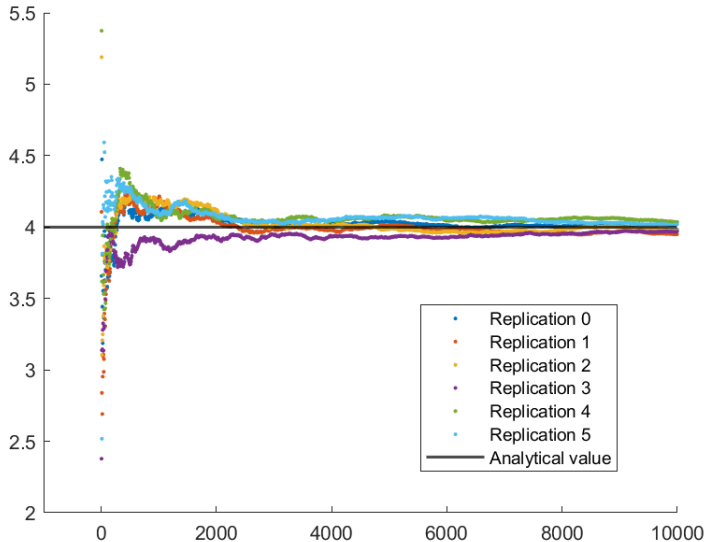
Steady-state system statistics are those statistics, if they exist, that are produced by simulating the operation of a **stationary** discrete-event system for an effectively **infinite** length of time.


In many systems, a steady state is not achieved until some time after the system is started or initiated. This initial situation is often identified as a transient state, start-up or warm-up period.

If a system is in a steady state, then the recently observed behavior of the system will continue into the future.

if the variables (called state variables) which define the behavior of the system or the process are unchanging in time.

Statistics Results





Grazie per l'attenzione!