Progetto di Simulazione di Sistemi

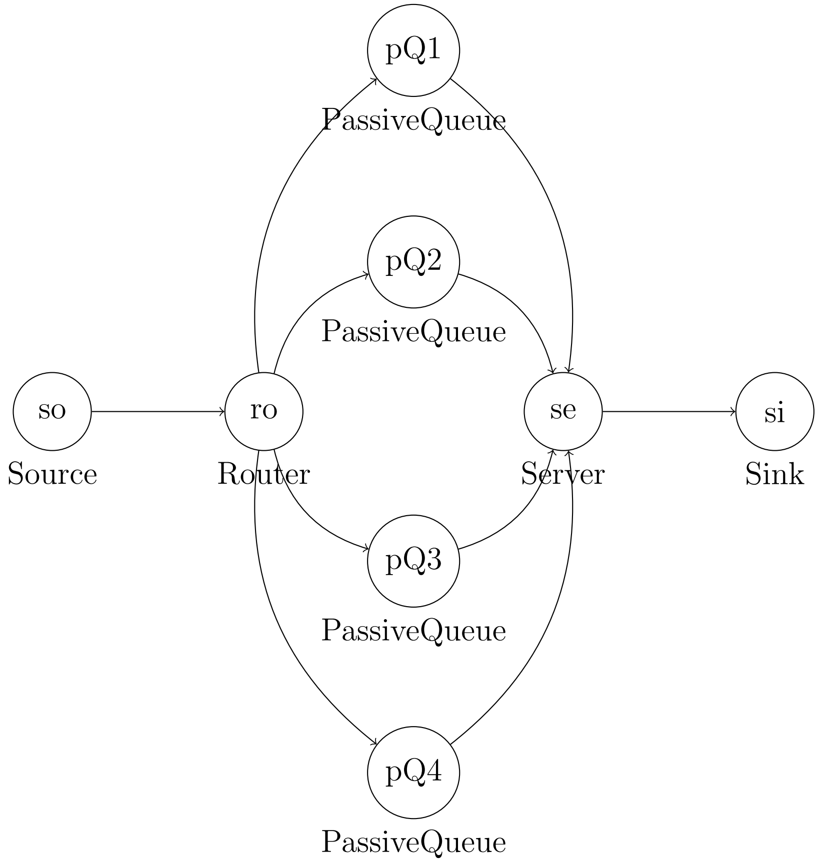
Samuele Evangelisti

a.a. 2019/2020

Contents

1 Modello

1.1 Rete



Il codice sorgente del file *network.ned* è riportato in appendice nella sezione A.1 a pagina 5.

1.2 Configurazione

Il codice sorgente del file *omnetpp.ini* è riportato in appendice nella sezione A.2 a pagina 7.

2 Misure di Prestazione

Le misure di prestazioni sulla quale ci si concentra sono le seguenti:

1. Mediana della distribuzione del tempo di risposta del sistema
2. Tempo minimo di permanenza dei *Job* nel sistema
3. Tempo massimo di permanenza dei *Job* nel sistema
4. Tempo medio di permanenza dei *Job* nel sistema
5. Numero medio di *Job* non serviti

Per ogni misura di prestazione vengono calcolati la stima puntuale e l'intervallo di confidenza al 95% e al 90%.
Per ottenere le misure di prestazione richieste sono state valutati i seguenti valori:

- *Network.sink.lifeTime:vector* (1, 2, 3, 4)
- *Netwoek.sink.lifeTime:max* (3)
- *Network.sink.lifeTime:mean* (4)
- *Network.server.droppedForDeadline:count* (5)

3 Risultati Sperimentali

3.1 Numero di Rilevazioni

Le rilevazioni di interesse vengono fornite dai *Job* che circolano nel sistema. Per ottenere un numero ragionevole di osservazioni è necessario che un numero ragionevole di *Job* vengano generati e terminati. Un valore ragionevole per la simulazione è di circa 1000 *Job*. Per ottenere questi valori per ogni run sono stati analizzati:

- *Network.source.created:last* (numero di *Job* creati)
- *Network.sink.lifeTime:vector* (numero di *Job* terminati)

Analizzando i risultati ottenuti (riportati in appendice nella sezione B.1) si ottiene

	$R_g(J)$	$R_t(J)$	
$J >= 1000$	828	828	86.25%
$900 <= J < 1000$	132	132	13.75%
$J < 900$	0	0	0.00%
	960		

Table 1: Analisi del numero di rilevazioni

con:

- $R_g(J)$: numero di run che hanno generato J *Job*
- $R_t(J)$: numero di run che hanno terminato J *Job*

A Codice Sorgente

Per l'implementazione delle funzionalità richieste dal modello di simulazione si è proceduto modificando alcune classi e alcuni moduli della libreria *queueinglib*. La nuova libreria è stata stata chiamata *pssqueueinglib* ed è stata utilizzata nel progetto al posto della libreria *queueinglib*.

Di seguito vengono riportate le modifiche apportate, nei moduli e nelle librerie il codice aggiunto è riportato sotto il commento

```
// progettoss
```

A.1 network.ned

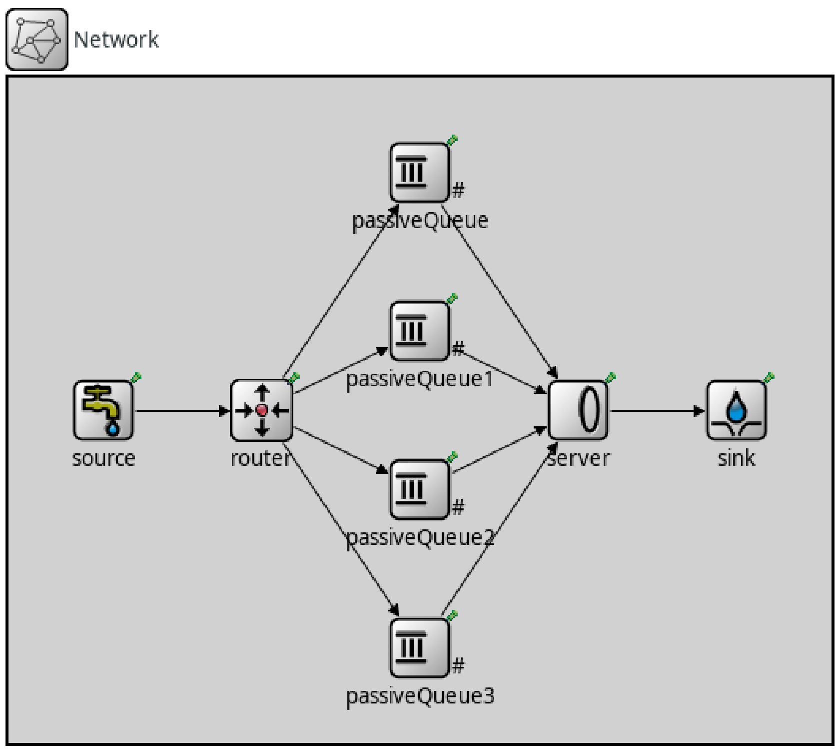


Figure 1: Visualizzazione grafica del file nework.ned

```
//
// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU Lesser General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU Lesser General Public License for more details.
//
// You should have received a copy of the GNU Lesser General Public License
// along with this program.  If not, see http://www.gnu.org/licenses/.
//
import org.omnetpp.queueing.PassiveQueue;
```

```
import org.omnetpp.queueing.Router;
import org.omnetpp.queueing.Server;
import org.omnetpp.queueing.Sink;
import org.omnetpp.queueing.Source;


//
// TODO documentation
//
network Network
{
    @display("bgb=520,420");
    submodules:
        source: Source {
            @display("p=60,210");
        }
        router: Router {
            @display("p=160,210");
        }
        passiveQueue: PassiveQueue {
            @display("p=260,60");
        }
        passiveQueue1: PassiveQueue {
            @display("p=260,160");
        }
        passiveQueue2: PassiveQueue {
            @display("p=260,260");
        }
        passiveQueue3: PassiveQueue {
            @display("p=260,360");
        }
        server: Server {
            @display("p=360,210");
        }
        sink: Sink {
            @display("p=460,210");
        }
    connections:
        source.out --> router.in++;
        router.out++ --> passiveQueue.in++;
        router.out++ --> passiveQueue1.in++;
        router.out++ --> passiveQueue2.in++;
        router.out++ --> passiveQueue3.in++;
        passiveQueue.out++ --> server.in++;
        passiveQueue1.out++ --> server.in++;
        passiveQueue2.out++ --> server.in++;
        passiveQueue3.out++ --> server.in++;
        server.out --> sink.in++;
}
```
Listing 1: "network.ned"

A.2 omnetpp.ini

```
[General]
network = Network

repeat = 20
sim-time-limit = 1000s

Network.router.routingAlgorithm = "pssRandom"
Network.server.fetchingAlgorithm = "exhaustiveService"
Network.server.checkJobDeadline = true

# [Example]
# Esponenziale di media 1/lambda = {0.5, 0.714285714286, 0.833333333333, 1}; lambda =
    {2.0, 1.4, 1.2, 1.0}
# Network.source.interArrivalTime = exponential(<1/lambda>s)

# H Uniforme su [a, b] = {[4.0, 6.0], [3.0, 7.0]}
# Network.source.jobRelativeDeadline = uniform(<a>s, <b>s)

# Code K = {1, 2, 4}
# Network.router.queueNumber = <K>

# S Esponenziale negativa di media 1/mu = {0.333333333333, 0.25}; mu = {3.0, 4.0}
# Network.server.serviceTime = exponential(<1/mu>s)

[Config n1lambda1H1K1mu1]

Network.source.interArrivalTime = exponential(0.5s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 1

Network.server.serviceTime = exponential(0.333333333333s)

[Config n2lambda1H1K1mu2]

Network.source.interArrivalTime = exponential(0.5s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 1

Network.server.serviceTime = exponential(0.25s)

[Config n3lambda1H1K2mu1]

Network.source.interArrivalTime = exponential(0.5s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 2

Network.server.serviceTime = exponential(0.333333333333s)

[Config n4lambda1H1K2mu2]

Network.source.interArrivalTime = exponential(0.5s)
```

```
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.25s)
[Config n5lambda1H1K3mu1]
Network.source.interArrivalTime = exponential(0.5s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.333333333333s)
[Config n6lambda1H1K3mu2]
Network.source.interArrivalTime = exponential(0.5s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.25s)
[Config n7lambda1H2K1mu1]
Network.source.interArrivalTime = exponential(0.5s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.333333333333s)
[Config n8lambda1H2K1mu2]
Network.source.interArrivalTime = exponential(0.5s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.25s)
[Config n9lambda1H2K2mu1]
Network.source.interArrivalTime = exponential(0.5s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.333333333333s)
[Config n10lambda1H2K2mu2]
```

```
Network.source.interArrivalTime = exponential(0.5s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.25s)
[Config n11lambda1H2K3mu1]
Network.source.interArrivalTime = exponential(0.5s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.333333333333s)
[Config n12lambda1H2K3mu2]
Network.source.interArrivalTime = exponential(0.5s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.25s)
[Config n13lambda2H1K1mu1]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.333333333333s)
[Config n14lambda2H1K1mu2]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.25s)
[Config n15lambda2H1K2mu1]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.333333333333s)
[Config n16lambda2H1K2mu2]
```

```
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.25s)
[Config n17lambda2H1K3mu1]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.333333333333s)
[Config n18lambda2H1K3mu2]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.25s)
[Config n19lambda2H2K1mu1]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.333333333333s)
[Config n20lambda2H2K1mu2]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.25s)
[Config n21lambda2H2K2mu1]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.333333333333s)
```

```
[Config n22lambda2H2K2mu2]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.25s)
[Config n23lambda2H2K3mu1]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.333333333333s)
[Config n24lambda2H2K3mu2]
Network.source.interArrivalTime = exponential(0.714285714286s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.25s)
[Config n25lambda3H1K1mu1]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.333333333333s)
[Config n26lambda3H1K1mu2]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.25s)
[Config n27lambda3H1K2mu1]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.333333333333s)
```

```
[Config n28lambda3H1K2mu2]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.25s)
[Config n29lambda3H1K3mu1]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.333333333333s)
[Config n30lambda3H1K3mu2]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.25s)
[Config n31lambda3H2K1mu1]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.333333333333s)
[Config n32lambda3H2K1mu2]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.25s)
[Config n33lambda3H2K2mu1]
Network.source.interArrivalTime = exponential(0.833333333333s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 2
```

```
Network.server.serviceTime = exponential(0.333333333333s)

[Config n34lambda3H2K2mu2]

Network.source.interArrivalTime = exponential(0.833333333333s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 2

Network.server.serviceTime = exponential(0.25s)

[Config n35lambda3H2K3mu1]

Network.source.interArrivalTime = exponential(0.833333333333s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 4

Network.server.serviceTime = exponential(0.333333333333s)

[Config n36lambda3H2K3mu2]

Network.source.interArrivalTime = exponential(0.833333333333s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 4

Network.server.serviceTime = exponential(0.25s)

[Config n37lambda4H1K1mu1]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 1

Network.server.serviceTime = exponential(0.333333333333s)

[Config n38lambda4H1K1mu2]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 1

Network.server.serviceTime = exponential(0.25s)

[Config n39lambda4H1K2mu1]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)

Network.router.queueNumber = 2
```

```
Network.server.serviceTime = exponential(0.333333333333s)
[Config n40lambda4H1K2mu2]
Network.source.interArrivalTime = exponential(1s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 2
Network.server.serviceTime = exponential(0.25s)
[Config n41lambda4H1K3mu1]
Network.source.interArrivalTime = exponential(1s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.333333333333s)
[Config n42lambda4H1K3mu2]
Network.source.interArrivalTime = exponential(1s)
Network.source.jobRelativeDeadline = uniform(4.0s, 6.0s)
Network.router.queueNumber = 4
Network.server.serviceTime = exponential(0.25s)
[Config n43lambda4H2K1mu1]
Network.source.interArrivalTime = exponential(1s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.333333333333s)
[Config n44lambda4H2K1mu2]
Network.source.interArrivalTime = exponential(1s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
Network.router.queueNumber = 1
Network.server.serviceTime = exponential(0.25s)
[Config n45lambda4H2K2mu1]
Network.source.interArrivalTime = exponential(1s)
Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)
```

```
Network.router.queueNumber = 2

Network.server.serviceTime = exponential(0.333333333333s)

[Config n46lambda4H2K2mu2]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 2

Network.server.serviceTime = exponential(0.25s)

[Config n47lambda4H2K3mu1]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 4

Network.server.serviceTime = exponential(0.333333333333s)

[Config n48lambda4H2K3mu2]

Network.source.interArrivalTime = exponential(1s)

Network.source.jobRelativeDeadline = uniform(3.0s, 7.0s)

Network.router.queueNumber = 4

Network.server.serviceTime = exponential(0.25s)
```

Listing 2: ”omnetpp.ini”

A.3 Job (pssqueueinglib)

Aggiunte:

- *simtime_t absoluteDeadline*: attributo che contiene la deadline assoluta del *Job (Job.h)*
- *void setAbsoluteDeadline(simtime_t absoluteDeadline)*: metodo per impostare la deadline assoluta del *Job (Job.h, Job.cc)*
- *simtime_t getAbsoluteDeadline()*: metodo per ottenere la deadline assoluta del *Job (Job.h, Job.cc)*

A.3.1 Job.h

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// `license' for details on this and other legal matters.
//

#ifndef __QUEUEING_JOB_H
#define __QUEUEING_JOB_H

#include <vector>
#include "Job_m.h"

namespace queueing {

class JobList;

/**
 * We extend the generated Job_Base class with support for split-join, as well
 * as the ability to enumerate all jobs in the system.
 *
 * To support split-join, Jobs manage parent-child relationships. A
 * relationship is created with the makeChildOf() or addChild() methods,
 * and lives until the parent or the child Job is destroyed.
 * It can be queried with the getParent() and getNumChildren()/getChild(k)
 * methods.
 *
 * To support enumerating all jobs in the system, each Job automatically
 * registers itself in a JobList module, if one exist in the model.
 * (If there's no JobList module, no registration takes place.) If there
 * are more than one JobList modules, the first one is chosen.
 * JobList can also be explicitly specified in the Job constructor.
 * The default JobList can be obtained with the JobList::getDefaultInstance()
 * method. Then one can query JobList for the set of Jobs currently present.
 */
class QUEUEING_API Job: public Job_Base
{
    friend class JobList;
    protected:
        Job *parent;
        std::vector<Job*> children;
        JobList *jobList;
        virtual void setParent(Job *parent); // only for addChild()
```

```
        virtual void parentDeleted();
        virtual void childDeleted(Job *child);
        // progettoss
        simtime_t absoluteDeadline;
    public:
        /**
         * Creates a job with the given name, message kind, and jobList. If
         * jobList==nullptr, the default one (or none if none exist) will be chosen.
         */
        Job(const char *name=nullptr, int kind=0, JobList *table=nullptr);

        /** Copy constructor */
        Job(const Job& job);

        /** Destructor */
        virtual ~Job();

        /** Duplicates this job */
        virtual Job *dup() const override {return new Job(*this);}

        /** Assignment operator. Does not affect parent, children and jobList. */
        Job& operator=(const Job& job);

        /** @name Parent-child relationships */
        //@{
        /** Returns the parent job. Returns nullptr if there's no parent or it no longer
             exists. */
        virtual Job *getParent();

        /** Returns the number of children. Deleted children are automatically removed
            from this list. */
        virtual int getNumChildren() const;

        /** Returns the kth child. Throws an error if index is out of range. */
        virtual Job *getChild(int k);

        /** Marks the given job as the child of this one. */
        void addChild(Job *child);

        /** Same as addChild(), but has to be invoked on the child job */
        virtual void makeChildOf(Job *parent);
        //@}

        /** Returns the JobList where this job has been registered. */
        JobList *getContainingJobList() {return jobList;}

        // progettoss
        void setAbsoluteDeadline(simtime_t absoluteDeadline);

        simtime_t getAbsoluteDeadline();

};

}; // namespace

#endif
```

A.3.2 Job.cc

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// `license' for details on this and other legal matters.
//

#include <algorithm>
#include "Job.h"
#include "JobList.h"

namespace queueing {

Job::Job(const char *name, int kind, JobList *jobList) : Job_Base(name, kind)
{
    parent = nullptr;
    if (jobList == nullptr && JobList::getDefaultInstance() != nullptr)
        jobList = JobList::getDefaultInstance();
    this->jobList = jobList;
    if (jobList != nullptr)
        jobList->registerJob(this);
}

Job::Job(const Job& job)
{
    setName(job.getName());
    operator=(job);
    parent = nullptr;
    jobList = job.jobList;
    if (jobList != nullptr)
        jobList->registerJob(this);
}

Job::~Job()
{
    if (parent)
        parent->childDeleted(this);
    for (int i = 0; i < (int)children.size(); i++)
        children[i]->parentDeleted();
    if (jobList != nullptr)
        jobList->deregisterJob(this);
}

Job& Job::operator=(const Job& job)
{
    if (this == &job)
        return *this;
    Job_Base::operator=(job);
    // leave parent and jobList untouched
    return *this;
}

Job *Job::getParent()
{
    return parent;
```

```
}

void Job::setParent(Job *parent)
{
    this->parent = parent;
}

int Job::getNumChildren() const
{
    return children.size();
}

Job *Job::getChild(int k)
{
    if (k < 0 || k >= (int)children.size())
        throw cRuntimeError(this, "child index %d out of bounds", k);
    return children[k];
}

void Job::makeChildOf(Job *parent)
{
    parent->addChild(this);
}

void Job::addChild(Job *child)
{
    child->setParent(this);
    ASSERT(std::find(children.begin(), children.end(), child) == children.end());
    children.push_back(child);
}

void Job::parentDeleted()
{
    parent = nullptr;
}

void Job::childDeleted(Job *child)
{
    std::vector<Job *>::iterator it = std::find(children.begin(), children.end(), child)
        ;
    ASSERT(it != children.end());
    children.erase(it);
}

void Job::setAbsoluteDeadline(simtime_t absoluteDeadline)
{
    this->absoluteDeadline = absoluteDeadline;
}

simtime_t Job::getAbsoluteDeadline()
{
    return absoluteDeadline;
}

}; // namespace
```

A.4 Source (pssqueueinglib)

Aggiunte:

- *double jobRelativeDeadline @unit(s) = default(0s)*: parametro per impostare la deadline relativa dei *Job (Source.ned)*

Modifiche:

- *Job *SourceBase::createJob()*: il *Job* viene configurato con la sua deadline assoluta *(Source.cc)*

A.4.1 Source.ned

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// 'license' for details on this and other legal matters.
//

package org.omnetpp.queueing;

//
// A module that generates jobs. One can specify the number of jobs to be generated,
// the starting and ending time, and interval between generating jobs.
// Job generation stops when the number of jobs or the end time has been reached,
// whichever occurs first. The name, type and priority of jobs can be set as well.
// One can specify the job relative deadline.
//
simple Source
{
    parameters:
        @group(Queueing);
        @signal[created](type="long");
        @statistic[created](title="the number of jobs created";record=last;
           interpolationmode=none);
        @display("i=block/source");
        int numJobs = default(-1);                // number of jobs to be generated (-1
           means no limit)
        volatile double interArrivalTime @unit(s); // time between generated jobs
        string jobName = default("job");          // the base name of the generated job (
           will be the module name if left empty)
        volatile int jobType = default(0);        // the type attribute of the created
           job (used by classifers and other modules)
        volatile int jobPriority = default(0);    // priority of the job
        double startTime @unit(s) = default(interArrivalTime); // when the module sends
           out the first job
        double stopTime @unit(s) = default(-1s); // when the module stops the job
           generation (-1 means no limit)
        // progettoss
        double jobRelativeDeadline @unit(s) = default(0s);    // job relative deadline
    gates:
        output out;
}
```

Listing 3: "Source.ned"

A.4.2 Source.cc

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// `license' for details on this and other legal matters.
//

#include "Source.h"
#include "Job.h"

namespace queueing {

void SourceBase::initialize()
{
    createdSignal = registerSignal("created");
    jobCounter = 0;
    WATCH(jobCounter);
    jobName = par("jobName").stringValue();
    if (jobName == "")
        jobName = getName();
}

Job *SourceBase::createJob()
{
    char buf[80];
    sprintf(buf, "%.60s-%d", jobName.c_str(), ++jobCounter);
    Job *job = new Job(buf);
    job->setKind(par("jobType"));
    job->setPriority(par("jobPriority"));
    job->setAbsoluteDeadline(simTime() + par("jobRelativeDeadline"));
    return job;
}

void SourceBase::finish()
{
    emit(createdSignal, jobCounter);
}

//----

Define_Module(Source);

void Source::initialize()
{
    SourceBase::initialize();
    startTime = par("startTime");
    stopTime = par("stopTime");
    numJobs = par("numJobs");

    // schedule the first message timer for start time
    scheduleAt(startTime, new cMessage("newJobTimer"));
}

void Source::handleMessage(cMessage *msg)
{
```

```
    ASSERT(msg->isSelfMessage());

    if ((numJobs < 0 || numJobs > jobCounter) && (stopTime < 0 || stopTime > simTime()))
        {
        // reschedule the timer for the next message
        scheduleAt(simTime() + par("interArrivalTime").doubleValue(), msg);

        Job *job = createJob();
        send(job, "out");
    }
    else {
        // finished
        delete msg;
    }
}

//----

Define_Module(SourceOnce);

void SourceOnce::initialize()
{
    SourceBase::initialize();
    simtime_t time = par("time");
    scheduleAt(time, new cMessage("newJobTimer"));
}

void SourceOnce::handleMessage(cMessage *msg)
{
    ASSERT(msg->isSelfMessage());
    delete msg;

    int n = par("numJobs");
    for (int i = 0; i < n; i++) {
        Job *job = createJob();
        send(job, "out");
    }
}

}; //namespace
```

Listing 4: "Source.cc"

A.5 Router (pssqueueinglib)

Aggiunte:

- *int queueNumber = default(sizeof(out)-1)*: numero di code da utilizzare *(Router.ned)*
- *ALG_PSSRANDOM*: algoritmo che consente di inoltrare i messaggi solo alle prime n code in maniera casuale *(Router.h)*
- *int queueNumber*: numero di code da utilizzare *(Router.h)*

Modifiche:

- *string routingAlgorithm @enum("random","roundRobin","shortestQueue","minDelay","pssRandom") = default("random")*: *"pssRandom"* permette di inoltrare i messaggi solo alle prime n code in maniera casuale *(Router.ned)*
- *void Router::initialize()*: inizializzazione dell'algoritmo di instradamento e del numero di code da utilizzare *(Router.cc)*
- *void Router::handleMessage(cMessage *msg)*: implementazione dell'algoritmo di instradamento *ALG_PSSRANDOM (Router.cc)*

A.5.1 Router.ned

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// 'license' for details on this and other legal matters.
//

package org.omnetpp.queueing;

//
// Sends the messages to different outputs depending on a set algorithm.
// Sends the messages to first queueNumber-th queues.
//
// @author rhornig, Samuele Evangelisti
// @todo minDelay not implemented
//
simple Router
{
    parameters:
        @group(Queueing);
        @display("i=block/routing");
        string routingAlgorithm @enum("random","roundRobin","shortestQueue","minDelay","
            pssRandom") = default("random");
        volatile int randomGateIndex = default(intuniform(0, sizeof(out)-1));    // the
            destination gate in case of random routing
        // progettoss
        int queueNumber = default(sizeof(out)-1);    // queue number limit
    gates:
        input in[];
        output out[];
```

```
}
```

Listing 5: "Router.ned"

A.5.2 Router.h

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// 'license' for details on this and other legal matters.
//

#ifndef __QUEUEING_ROUTER_H
#define __QUEUEING_ROUTER_H

#include "QueueingDefs.h"

namespace queueing {

// routing algorithms
enum {
    ALG_RANDOM,
    ALG_ROUND_ROBIN,
    ALG_MIN_QUEUE_LENGTH,
    ALG_MIN_DELAY,
    ALG_MIN_SERVICE_TIME,
    // progettoss
    ALG_PSSRANDOM
};

/**
 * Sends the messages to different outputs depending on a set algorithm.
 * Sends the messages to first queueNumber-th queues.
 */
class QUEUEING_API Router : public cSimpleModule
{
    private:
        int routingAlgorithm;  // the algorithm we are using for routing
        int rrCounter;         // msgCounter for round robin routing
        // progettoss
        int queueNumber;
    protected:
        virtual void initialize() override;
        virtual void handleMessage(cMessage *msg) override;
};

}; //namespace

#endif
```

Listing 6: "Router.h"

A.5.3 Router.cc

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// `license' for details on this and other legal matters.
//

#include "Router.h"

namespace queueing {

Define_Module(Router);

void Router::initialize()
{
    const char *algName = par("routingAlgorithm");
    if (strcmp(algName, "random") == 0) {
        routingAlgorithm = ALG_RANDOM;
    }
    else if (strcmp(algName, "roundRobin") == 0) {
        routingAlgorithm = ALG_ROUND_ROBIN;
    }
    else if (strcmp(algName, "minQueueLength") == 0) {
        routingAlgorithm = ALG_MIN_QUEUE_LENGTH;
    }
    else if (strcmp(algName, "minDelay") == 0) {
        routingAlgorithm = ALG_MIN_DELAY;
    }
    else if (strcmp(algName, "minServiceTime") == 0) {
        routingAlgorithm = ALG_MIN_SERVICE_TIME;
    }
    else if (strcmp(algName, "pssRandom") == 0) {
        routingAlgorithm = ALG_PSSRANDOM;
    }
    rrCounter = 0;
    int qn = par("queueNumber").intValue() - 1;
    if (qn < 0 || qn > gateSize("out") - 1)
        throw cRuntimeError("Invalid queue number");
    else
        queueNumber = qn;
}

void Router::handleMessage(cMessage *msg)
{
    int outGateIndex = -1;  // by default we drop the message

    switch (routingAlgorithm) {
        case ALG_RANDOM:
            outGateIndex = par("randomGateIndex");
            break;

        case ALG_ROUND_ROBIN:
            outGateIndex = rrCounter;
            rrCounter = (rrCounter + 1) % gateSize("out");
            break;

```

```
        case ALG_MIN_QUEUE_LENGTH:
            // TODO implementation missing
            outGateIndex = -1;
            break;

        case ALG_MIN_DELAY:
            // TODO implementation missing
            outGateIndex = -1;
            break;

        case ALG_MIN_SERVICE_TIME:
            // TODO implementation missing
            outGateIndex = -1;
            break;

        case ALG_PSSRANDOM:
            outGateIndex = intuniform(0, queueNumber);
            break;

        default:
            outGateIndex = -1;
            break;
    }

    // send out if the index is legal
    if (outGateIndex < 0 || outGateIndex >= gateSize("out"))
        throw cRuntimeError("Invalid output gate selected during routing");

    send(msg, "out", outGateIndex);
}

}; //namespace
```
Listing 7: "Router.cc"

A.6 SelectionStrategies (pssqueueinglib)

Aggiunte:

- *class QUEUEING_API ExhaustiveServiceSelectionStrategy : public SelectionStrategy*: implementa la *SelectionStrategy* per ottenere un exhaustive service *(SelectionStrategies.h, SelectionStrategies.cc)*

Modifiche:

- *SelectionStrategy *SelectionStrategy::create(const char *algName, cSimpleModule *module, bool selectOnInGate*: inizializzazione di *ExhaustiveServieSelectionStrategy (SelectionStrategies.cc)*

A.6.1 SelectionStrategies.h

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// `license' for details on this and other legal matters.
//

#ifndef __QUEUEING_SELECTIONSTRATEGIES_H
#define __QUEUEING_SELECTIONSTRATEGIES_H

#include "QueueingDefs.h"

namespace queueing {

/**
 * Selection strategies used in queue, server and router classes to decide
 * which module to choose for further interaction.
 */
class QUEUEING_API SelectionStrategy : public cObject
{
    protected:
        bool isInputGate;
        int gateSize;          // the size of the gate vector
        cModule *hostModule; // the module using the strategy
    public:
        // on which module's gates should be used for selection
        // if selectOnInGate is true, then we will use "in" gate otherwise "out" is used
        SelectionStrategy(cSimpleModule *module, bool selectOnInGate);
        virtual ~SelectionStrategy();

        static SelectionStrategy * create(const char *algName, cSimpleModule *module,
            bool selectOnInGate);

        // which gate index the selection strategy selected
        virtual int select() = 0;
        // returns the i-th module's gate which connects to our host module
        cGate *selectableGate(int i);
    protected:
        // is this module selectable according to the policy? (queue is selectable if
            not empty, server is selectable if idle)
        virtual bool isSelectable(cModule *module);
};
```

```
/**
 * Priority based selection. The first selectable index will be returned.
 */
class QUEUEING_API PrioritySelectionStrategy : public SelectionStrategy
{
    public:
        PrioritySelectionStrategy(cSimpleModule *module, bool selectOnInGate);
        virtual int select() override;
};

/**
 * Random selection from the selectable modules, with uniform distribution.
 */
class QUEUEING_API RandomSelectionStrategy : public SelectionStrategy
{
    public:
        RandomSelectionStrategy(cSimpleModule *module, bool selectOnInGate);
        virtual int select() override;
};

/**
 * Uses Round Robin selection, but skips any module that is not available currently.
 */
class QUEUEING_API RoundRobinSelectionStrategy : public SelectionStrategy
{
    protected:
        int lastIndex; // the index of the module last time used
    public:
        RoundRobinSelectionStrategy(cSimpleModule *module, bool selectOnInGate);
        virtual int select() override;
};

/**
 * Chooses the shortest queue. If there are more than one
 * with the same length, it chooses by priority among them.
 * This strategy is for output only (i.e. for router module).
 */
class QUEUEING_API ShortestQueueSelectionStrategy : public SelectionStrategy
{
    public:
        ShortestQueueSelectionStrategy(cSimpleModule *module, bool selectOnInGate);
        virtual int select() override;
};

/**
 * Chooses the longest queue (where length>0 of course).
 * Input strategy (for servers).
 */
class QUEUEING_API LongestQueueSelectionStrategy : public SelectionStrategy
{
    public:
        LongestQueueSelectionStrategy(cSimpleModule *module, bool selectOnInGate);
        virtual int select() override;
};

// progettoss
/**
 * End all the tasks in a queue, then chooses cyclically the next one.
```

```
 * Input strategy (for servers).
 */
class QUEUEING_API ExhaustiveServiceSelectionStrategy : public SelectionStrategy
{
    private:
        int actualInputGate;    // actual input gate
    public:
        ExhaustiveServiceSelectionStrategy(cSimpleModule *module, bool selectOnInGate);
        virtual int select() override;
};

}; //namespace

#endif
```

Listing 8: "SelectionStrategies.h"

A.6.2 SelectionStrategies.cc

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// `license' for details on this and other legal matters.
//

#include "SelectionStrategies.h"
#include "PassiveQueue.h"
#include "Server.h"

namespace queueing {

SelectionStrategy::SelectionStrategy(cSimpleModule *module, bool selectOnInGate)
{
    hostModule = module;
    isInputGate = selectOnInGate;
    gateSize = isInputGate ? hostModule->gateSize("in") : hostModule->gateSize("out");
}

SelectionStrategy::~SelectionStrategy()
{
}

SelectionStrategy *SelectionStrategy::create(const char *algName, cSimpleModule *module,
    bool selectOnInGate)
{
    SelectionStrategy *strategy = nullptr;

    if (strcmp(algName, "priority") == 0) {
        strategy = new PrioritySelectionStrategy(module, selectOnInGate);
    }
    else if (strcmp(algName, "random") == 0) {
        strategy = new RandomSelectionStrategy(module, selectOnInGate);
    }
    else if (strcmp(algName, "roundRobin") == 0) {
        strategy = new RoundRobinSelectionStrategy(module, selectOnInGate);
```

```
    }
    else if (strcmp(algName, "shortestQueue") == 0) {
        strategy = new ShortestQueueSelectionStrategy(module, selectOnInGate);
    }
    else if (strcmp(algName, "longestQueue") == 0) {
        strategy = new LongestQueueSelectionStrategy(module, selectOnInGate);
    }
    else if (strcmp(algName, "exhaustiveService") == 0) {
        strategy = new ExhaustiveServiceSelectionStrategy(module, selectOnInGate);
    }

    return strategy;
}

cGate *SelectionStrategy::selectableGate(int i)
{
    if (isInputGate)
        return hostModule->gate("in", i)->getPreviousGate();
    else
        return hostModule->gate("out", i)->getNextGate();
}

bool SelectionStrategy::isSelectable(cModule *module)
{
    if (isInputGate) {
        IPassiveQueue *pqueue = dynamic_cast<IPassiveQueue *>(module);
        if (pqueue != nullptr)
            return pqueue->length() > 0;
    }
    else {
        IServer *server = dynamic_cast<IServer *>(module);
        if (server != nullptr)
            return server->isIdle();
    }

    throw cRuntimeError("Only IPassiveQueue (as input) and IServer (as output) is 
       supported by this Strategy");
}

//
   ----------------------------------------------------------------------------------------


PrioritySelectionStrategy::PrioritySelectionStrategy(cSimpleModule *module, bool
    selectOnInGate) :
    SelectionStrategy(module, selectOnInGate)
{
}

int PrioritySelectionStrategy::select()
{
    // return the smallest selectable index
    for (int i = 0; i < gateSize; i++)
        if (isSelectable(selectableGate(i)->getOwnerModule()))
            return i;

    // if none of them is selectable return an invalid no.
    return -1;
}
```

```
// ----------------------------------------------------------------------------------------------------------------

RandomSelectionStrategy::RandomSelectionStrategy(cSimpleModule *module, bool
    selectOnInGate) :
     SelectionStrategy(module, selectOnInGate)
{
}

int RandomSelectionStrategy::select()
{
     // return the smallest selectable index
     int noOfSelectables = 0;
     for (int i = 0; i < gateSize; i++)
          if (isSelectable(selectableGate(i)->getOwnerModule()))
               noOfSelectables++;

     int rnd = hostModule->intuniform(1, noOfSelectables);

     for (int i = 0; i < gateSize; i++)
          if (isSelectable(selectableGate(i)->getOwnerModule()) && (--rnd == 0))
               return i;

     return -1;
}

// ----------------------------------------------------------------------------------------------------------------

RoundRobinSelectionStrategy::RoundRobinSelectionStrategy(cSimpleModule *module, bool
    selectOnInGate) :
     SelectionStrategy(module, selectOnInGate)
{
     lastIndex = -1;
}

int RoundRobinSelectionStrategy::select()
{
     // return the smallest selectable index
     for (int i = 0; i < gateSize; ++i) {
          lastIndex = (lastIndex+1) % gateSize;
          if (isSelectable(selectableGate(lastIndex)->getOwnerModule()))
               return lastIndex;
     }

     // if none of them is selectable return an invalid no.
     return -1;
}

// ----------------------------------------------------------------------------------------------------------------

ShortestQueueSelectionStrategy::ShortestQueueSelectionStrategy(cSimpleModule *module,
    bool selectOnInGate) :
     SelectionStrategy(module, selectOnInGate)
```

```
{
}

int ShortestQueueSelectionStrategy::select()
{
    // return the smallest selectable index
    int result = -1;  // by default none of them is selectable
    int sizeMin = INT_MAX;
    for (int i = 0; i < gateSize; ++i) {
        cModule *module = selectableGate(i)->getOwnerModule();
        int length = (check_and_cast<IPassiveQueue *>(module))->length();
        if (isSelectable(module) && (length < sizeMin)) {
            sizeMin = length;
            result = i;
        }
    }
    return result;
}

// ------------------------------------------------------------------------------------------------

LongestQueueSelectionStrategy::LongestQueueSelectionStrategy(cSimpleModule *module, bool
     selectOnInGate) :
    SelectionStrategy(module, selectOnInGate)
{
}

int LongestQueueSelectionStrategy::select()
{
    // return the longest selectable queue
    int result = -1;  // by default none of them is selectable
    int sizeMax = -1;
    for (int i = 0; i < gateSize; ++i) {
        cModule *module = selectableGate(i)->getOwnerModule();
        int length = (check_and_cast<IPassiveQueue *>(module))->length();
        if (isSelectable(module) && length > sizeMax) {
            sizeMax = length;
            result = i;
        }
    }
    return result;
}

// ------------------------------------------------------------------------------------------------

ExhaustiveServiceSelectionStrategy::ExhaustiveServiceSelectionStrategy(cSimpleModule *
    module, bool selectOnInGate) :
    SelectionStrategy(module, selectOnInGate)
{
    actualInputGate = 0;
}

int ExhaustiveServiceSelectionStrategy::select()
{
    // previously selected queue is not empty
```

```
    if (isSelectable(selectableGate(actualInputGate)->getOwnerModule()))
        return actualInputGate;
    // scan cyclically the next non empty queue
    else {
        for (int i = 1; i < gateSize; i++) {
            int gn = (actualInputGate + i) % gateSize;
            if (isSelectable(selectableGate(gn)->getOwnerModule())) {
                actualInputGate = gn;
                return gn;
            }
        }
    }

    // if none of them is selectable return an invalid no.
    return -1;
}

}; //namespace
```

Listing 9: "SelectionStrategies.cc"

A.7 Server (pssqueueinglib)

Aggiunte:

- *@signal[droppedForDeadline](type=”long”)*: *signal* per la registrazione dei *Job* scartati a causa di un inizio di servizio successivo alla loro *absoluteDeadline (Server.ned)*
- *@statistic[droppedForDeadline](title=”drop event for deadline reached”;record=vector?,count;interpolationm* statistica relativa ai *Job* scartati a causa di un inizio di servizio successivo alla loro *absoluteDeadline (Server.ned)*
- *bool checkJobDeadline = default(true)*: specifica se sia necessario controllare *Job.absoluteDeadline* prima dell'inizio del servizio *(Server.ned)*
- *simsignal_t droppedForDeadlineSignal*: *signal* per la registrazione dei *Job* scartati a causa di un inizio di servizio successivo alla loro *absoluteDeadline (Server.h)*
- *bool checkJobDeadline*: specifica se sia necessario controllare *Job.absoluteDeadline* prima dell'inizio del servizio *(Server.h)*

Modifiche:

- *void Server::initialize()*: configurazione dei nuovi valori aggiunti *(Server.cc)*
- *void Server::handleMessage(cMessage *msg)*: modifiche al metodo per implementare il controllo di *Job.absoluteDeadline* e richiedere un nuovo *Job* in caso quello attuale venga scartato

A.7.1 Server.ned

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// `license' for details on this and other legal matters.
//

package org.omnetpp.queueing;

//
// Queue server. It serves multiple input queues (PassiveQueue), using a preset
// algorithm. Inputs must be connected to Passive Queues (PassiveQueue)
//
simple Server
{
    parameters:
        @group(Queueing);
        @display("i=block/server");
        @signal[busy](type="bool");
        @statistic[busy](title="server busy state";record=vector?,timeavg;
            interpolationmode=sample-hold);
        // progettoss
        @signal[droppedForDeadline](type="long");
        @statistic[droppedForDeadline](title="drop event for deadline reached";record=
            vector?,count;interpolationmode=none);
```

```
        string fetchingAlgorithm @enum("priority","random","roundRobin","longestQueue","
            exhaustiveService") = default("priority");
              // how the next job will be choosen from the attached queues
        volatile double serviceTime @unit(s); // service time of a job
        // progettoss
        bool checkJobDeadline = default(false);
    gates:
        input in[];
        output out;
}
```

Listing 10: "Server.ned"

A.7.2 Server.h

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// `license' for details on this and other legal matters.
//

#ifndef __QUEUEING_SERVER_H
#define __QUEUEING_SERVER_H

#include "IServer.h"

namespace queueing {

class Job;
class SelectionStrategy;

/**
 * The queue server. It cooperates with several Queues that which queue up
 * the jobs, and send them to Server on request.
 *
 * @see PassiveQueue
 */
class QUEUEING_API Server : public cSimpleModule, public IServer
{
    private:
        simsignal_t busySignal;
        bool allocated;

        SelectionStrategy *selectionStrategy;

        Job *jobServiced;
        cMessage *endServiceMsg;

        // progettoss
        simsignal_t droppedForDeadlineSignal;
        bool checkJobDeadline;

    public:
        Server();
        virtual ~Server();
```

```
    protected:
        virtual void initialize() override;
        virtual int numInitStages() const override {return 2;}
        virtual void handleMessage(cMessage *msg) override;
        virtual void refreshDisplay() const override;
        virtual void finish() override;

    public:
        virtual bool isIdle() override;
        virtual void allocate() override;
};

}; //namespace

#endif
```

Listing 11: "Server.h"

A.7.3 Server.cc

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
// Copyright (C) 2006-2015 OpenSim Ltd.
//
// This file is distributed WITHOUT ANY WARRANTY. See the file
// `license' for details on this and other legal matters.
//

#include "Server.h"
#include "Job.h"
#include "SelectionStrategies.h"
#include "IPassiveQueue.h"

namespace queueing {

Define_Module(Server);

Server::Server()
{
    selectionStrategy = nullptr;
    jobServiced = nullptr;
    endServiceMsg = nullptr;
    allocated = false;
}

Server::~Server()
{
    delete selectionStrategy;
    delete jobServiced;
    cancelAndDelete(endServiceMsg);
}

void Server::initialize()
{
    busySignal = registerSignal("busy");
    emit(busySignal, false);
```

```
    endServiceMsg = new cMessage("end-service");
    jobServiced = nullptr;
    allocated = false;
    selectionStrategy = SelectionStrategy::create(par("fetchingAlgorithm"), this, true);
    if (!selectionStrategy)
        throw cRuntimeError("invalid selection strategy");
    droppedForDeadlineSignal = registerSignal("droppedForDeadline");
    checkJobDeadline = par("checkJobDeadline").boolValue();
}

void Server::handleMessage(cMessage *msg)
{
    if (msg == endServiceMsg) {
        ASSERT(jobServiced != nullptr);
        ASSERT(allocated);
        simtime_t d = simTime() - endServiceMsg->getSendingTime();
        jobServiced->setTotalServiceTime(jobServiced->getTotalServiceTime() + d);
        send(jobServiced, "out");
        jobServiced = nullptr;
        allocated = false;
        emit(busySignal, false);

        // examine all input queues, and request a new job from a non empty queue
        int k = selectionStrategy->select();
        if (k >= 0) {
            EV << "requesting job from queue " << k << endl;
            cGate *gate = selectionStrategy->selectableGate(k);
            check_and_cast<IPassiveQueue *>(gate->getOwnerModule())->request(gate->
                getIndex());
        }
    }
    else {
        if (!allocated)
            error("job arrived, but the sender did not call allocate() previously");
        if (jobServiced)
            throw cRuntimeError("a new job arrived while already servicing one");
        jobServiced = check_and_cast<Job *>(msg);
        simtime_t serviceTime = par("serviceTime");
        if (checkJobDeadline) {
            if (jobServiced->getAbsoluteDeadline() < simTime()) {
                EV << "Dropped!" << endl;
                if (hasGUI())
                    bubble("Dropped!");
                emit(droppedForDeadlineSignal, 1);
                delete msg;
                allocated = false;
                jobServiced = nullptr;
                int k = selectionStrategy->select();
                if (k >= 0) {
                    EV << "requesting job from queue " << k << endl;
                    cGate *gate = selectionStrategy->selectableGate(k);
                    check_and_cast<IPassiveQueue *>(gate->getOwnerModule())->request(
                        gate->getIndex());
                }
            }
            else {
                scheduleAt(simTime()+serviceTime, endServiceMsg);
                emit(busySignal, true);
```

```
            }
        }
        else {
            scheduleAt(simTime()+serviceTime, endServiceMsg);
            emit(busySignal, true);
        }
    }
}

void Server::refreshDisplay() const
{
    getDisplayString().setTagArg("i2", 0, jobServiced ? "status/execute" : "");
}

void Server::finish()
{
}

bool Server::isIdle()
{
    return !allocated;  // we are idle if nobody has allocated us for processing
}

void Server::allocate()
{
    allocated = true;
}

}; //namespace
```

Listing 12: "Server.cc"

B Risultati Sperimentali

B.1 Numero di Rilevazioni

B.1.1 Job Generati

	Configurazione 1									
Run 1-10	2002	2036	1988	2107	1971	1934	2001	2004	1997	2071
Run 11-20	2050	1950	2013	1981	1940	1961	1955	1979	2016	2003

Table 2: Numero di rilevamenti nella configurazione 1

	Configurazione 2									
Run 1-10	1968	2026	2026	1983	1982	1973	1938	2033	2063	2025
Run 11-20	2011	1970	2014	1966	1949	1970	1923	1966	1951	2030

Table 3: Numero di rilevamenti nella configurazione 2

	Configurazione 3									
Run 1-10	2003	2027	1981	2110	1956	1937	1984	2009	1997	2051
Run 11-20	2051	1958	2010	1982	1941	1955	1972	1984	2022	1988

Table 4: Numero di rilevamenti nella configurazione 3

	Configurazione 4									
Run 1-10	1970	2026	2026	1983	1982	1973	1938	2033	2063	2025
Run 11-20	2011	1970	2014	1964	1948	1970	1923	1966	1951	2021

Table 5: Numero di rilevamenti nella configurazione 4

	Configurazione 5									
Run 1-10	2030	2009	1995	2079	1956	1931	1995	2008	2009	2057
Run 11-20	2050	1943	2004	1978	1943	1973	1955	1982	2024	1988

Table 6: Numero di rilevamenti nella configurazione 5

	Configurazione 6									
Run 1-10	1968	2026	2026	1983	1982	1973	1938	2033	2063	2025
Run 11-20	2011	1970	2017	1963	1949	1970	1923	1968	1951	2030

Table 7: Numero di rilevamenti nella configurazione 6

	Configurazione 7									
Run 1-10	2003	2025	1996	2107	1971	1934	1994	1999	2009	2060
Run 11-20	2049	1950	2008	1981	1940	1973	1966	1988	2022	2007

Table 8: Numero di rilevamenti nella configurazione 7

	Configurazione 8									
Run 1-10	1968	2026	2026	1983	1982	1973	1938	2033	2063	2025
Run 11-20	2011	1970	2014	1966	1949	1970	1923	1966	1951	2030

Table 9: Numero di rilevamenti nella configurazione 8

	Configurazione 9									
Run 1-10	2000	2015	1990	2112	1961	1933	1987	2013	1994	2056
Run 11-20	2055	1950	2014	1976	1940	1968	1959	1988	2034	1980

Table 10: Numero di rilevamenti nella configurazione 9

	Configurazione 10									
Run 1-10	1970	2026	2026	1983	1982	1973	1938	2033	2063	2022
Run 11-20	2011	1970	2014	1964	1949	1973	1923	1966	1955	2021

Table 11: Numero di rilevamenti nella configurazione 10

	Configurazione 11									
Run 1-10	2017	2040	1986	2089	1962	1931	1992	2025	2018	2070
Run 11-20	2051	1953	1992	1960	1943	1969	1952	1992	2044	1980

Table 12: Numero di rilevamenti nella configurazione 11

	Configurazione 12									
Run 1-10	1968	2026	2026	1983	1982	1973	1938	2033	2063	2024
Run 11-20	2011	1970	2018	1964	1949	1970	1923	1966	1955	2030

Table 13: Numero di rilevamenti nella configurazione 12

	Configurazione 13									
Run 1-10	1409	1386	1420	1386	1413	1339	1383	1429	1405	1434
Run 11-20	1430	1372	1415	1422	1355	1371	1359	1365	1416	1402

Table 14: Numero di rilevamenti nella configurazione 13

	Configurazione 14									
Run 1-10	1393	1386	1400	1401	1386	1326	1415	1452	1394	1418
Run 11-20	1495	1380	1415	1439	1338	1361	1331	1355	1410	1395

Table 15: Numero di rilevamenti nella configurazione 14

	Configurazione 15									
Run 1-10	1409	1386	1420	1386	1413	1339	1383	1430	1405	1433
Run 11-20	1430	1372	1415	1421	1355	1371	1361	1365	1412	1403

Table 16: Numero di rilevamenti nella configurazione 15

	Configurazione 16									
Run 1-10	1393	1386	1400	1401	1386	1326	1415	1452	1394	1418
Run 11-20	1495	1380	1415	1439	1338	1361	1331	1355	1410	1395

Table 17: Numero di rilevamenti nella configurazione 16

	Configurazione 17									
Run 1-10	1409	1387	1420	1386	1413	1339	1383	1427	1405	1434
Run 11-20	1430	1372	1415	1423	1355	1371	1355	1362	1415	1404

Table 18: Numero di rilevamenti nella configurazione 17

	Configurazione 18									
Run 1-10	1393	1386	1400	1401	1386	1326	1415	1452	1394	1418
Run 11-20	1495	1380	1415	1439	1338	1361	1331	1355	1410	1395

Table 19: Numero di rilevamenti nella configurazione 18

	Configurazione 19									
Run 1-10	1409	1386	1420	1386	1413	1339	1383	1427	1405	1434
Run 11-20	1430	1372	1415	1414	1355	1371	1359	1365	1410	1402

Table 20: Numero di rilevamenti nella configurazione 19

	Configurazione 20									
Run 1-10	1393	1386	1400	1401	1386	1326	1415	1452	1394	1418
Run 11-20	1495	1380	1415	1439	1338	1361	1331	1355	1410	1395

Table 21: Numero di rilevamenti nella configurazione 20

	Configurazione 21									
Run 1-10	1409	1386	1420	1386	1414	1339	1383	1430	1405	1435
Run 11-20	1430	1372	1415	1424	1355	1371	1361	1364	1416	1405

Table 22: Numero di rilevamenti nella configurazione 21

	Configurazione 22									
Run 1-10	1393	1386	1400	1401	1386	1326	1415	1452	1394	1418
Run 11-20	1495	1380	1415	1439	1338	1357	1331	1355	1410	1395

Table 23: Numero di rilevamenti nella configurazione 22

	Configurazione 23									
Run 1-10	1409	1387	1420	1386	1413	1339	1383	1427	1405	1433
Run 11-20	1430	1372	1415	1421	1355	1373	1359	1364	1414	1404

Table 24: Numero di rilevamenti nella configurazione 23

	Configurazione 24									
Run 1-10	1393	1386	1400	1401	1386	1326	1415	1452	1394	1418
Run 11-20	1495	1380	1415	1439	1338	1361	1331	1355	1410	1395

Table 25: Numero di rilevamenti nella configurazione 24

	Configurazione 25									
Run 1-10	1178	1204	1227	1195	1202	1132	1204	1246	1170	1232
Run 11-20	1259	1210	1203	1254	1163	1172	1129	1190	1232	1211

Table 26: Numero di rilevamenti nella configurazione 25

	Configurazione 26									
Run 1-10	1191	1197	1199	1195	1202	1137	1188	1243	1171	1213
Run 11-20	1280	1202	1183	1239	1137	1163	1112	1146	1242	1210

Table 27: Numero di rilevamenti nella configurazione 26

	Configurazione 27									
Run 1-10	1178	1204	1227	1195	1202	1132	1204	1246	1170	1232
Run 11-20	1259	1210	1203	1254	1163	1172	1129	1190	1232	1212

Table 28: Numero di rilevamenti nella configurazione 27

	Configurazione 28									
Run 1-10	1191	1197	1199	1195	1202	1137	1188	1243	1171	1213
Run 11-20	1280	1202	1183	1239	1137	1163	1112	1146	1242	1210

Table 29: Numero di rilevamenti nella configurazione 28

	Configurazione 29									
Run 1-10	1178	1204	1227	1195	1202	1132	1204	1246	1170	1232
Run 11-20	1259	1208	1203	1257	1163	1172	1129	1190	1232	1214

Table 30: Numero di rilevamenti nella configurazione 29

	Configurazione 30									
Run 1-10	1191	1197	1199	1195	1202	1137	1188	1243	1171	1213
Run 11-20	1280	1202	1183	1239	1137	1163	1112	1146	1242	1210

Table 31: Numero di rilevamenti nella configurazione 30

	Configurazione 31									
Run 1-10	1178	1204	1227	1195	1202	1132	1204	1245	1170	1232
Run 11-20	1259	1210	1203	1248	1163	1172	1129	1190	1232	1211

Table 32: Numero di rilevamenti nella configurazione 31

	Configurazione 32									
Run 1-10	1191	1197	1199	1195	1202	1137	1188	1243	1171	1213
Run 11-20	1280	1202	1183	1239	1137	1163	1112	1146	1242	1210

Table 33: Numero di rilevamenti nella configurazione 32

	Configurazione 33									
Run 1-10	1178	1204	1227	1195	1202	1132	1204	1246	1170	1232
Run 11-20	1259	1210	1203	1254	1163	1172	1129	1190	1229	1212

Table 34: Numero di rilevamenti nella configurazione 33

	Configurazione 34									
Run 1-10	1191	1197	1199	1195	1202	1137	1188	1243	1171	1213
Run 11-20	1280	1202	1183	1239	1137	1163	1112	1146	1242	1210

Table 35: Numero di rilevamenti nella configurazione 34

	Configurazione 35									
Run 1-10	1178	1204	1227	1195	1202	1132	1204	1246	1170	1233
Run 11-20	1259	1210	1203	1254	1163	1172	1129	1190	1241	1209

Table 36: Numero di rilevamenti nella configurazione 35

	Configurazione 36									
Run 1-10	1191	1197	1199	1195	1202	1137	1188	1243	1171	1213
Run 11-20	1280	1202	1183	1239	1137	1163	1112	1146	1242	1210

Table 37: Numero di rilevamenti nella configurazione 36

	Configurazione 37									
Run 1-10	1002	1004	1015	972	990	956	987	1042	987	1036
Run 11-20	1081	1014	972	1035	932	948	937	981	1048	982

Table 38: Numero di rilevamenti nella configurazione 37

	Configurazione 38									
Run 1-10	991	996	1007	970	1006	953	990	1042	989	1011
Run 11-20	1073	1018	976	1056	934	983	959	987	1050	1011

Table 39: Numero di rilevamenti nella configurazione 38

	Configurazione 39									
Run 1-10	1002	1004	1015	972	990	956	987	1042	987	1036
Run 11-20	1081	1014	972	1035	932	948	937	981	1048	982

Table 40: Numero di rilevamenti nella configurazione 39

	Configurazione 40									
Run 1-10	991	996	1007	970	1006	953	990	1042	989	1011
Run 11-20	1073	1018	976	1056	934	983	959	987	1050	1011

Table 41: Numero di rilevamenti nella configurazione 40

	Configurazione 41									
Run 1-10	1002	1004	1015	972	990	956	987	1042	987	1036
Run 11-20	1081	1014	972	1035	932	948	937	981	1048	982

Table 42: Numero di rilevamenti nella configurazione 41

	Configurazione 42									
Run 1-10	991	996	1007	970	1006	953	990	1042	989	1011
Run 11-20	1073	1018	976	1056	934	983	959	987	1050	1011

Table 43: Numero di rilevamenti nella configurazione 42

	Configurazione 43									
Run 1-10	1002	1004	1015	972	990	956	987	1042	987	1036
Run 11-20	1081	1014	972	1035	932	948	937	981	1049	982

Table 44: Numero di rilevamenti nella configurazione 43

	Configurazione 44									
Run 1-10	991	996	1007	970	1006	953	990	1042	989	1011
Run 11-20	1073	1018	976	1056	934	983	959	987	1050	1011

Table 45: Numero di rilevamenti nella configurazione 44

	Configurazione 45									
Run 1-10	1002	1004	1015	972	990	956	987	1042	987	1036
Run 11-20	1081	1014	972	1035	932	948	937	981	1049	982

Table 46: Numero di rilevamenti nella configurazione 45

	Configurazione 46									
Run 1-10	991	996	1007	970	1006	953	990	1042	989	1011
Run 11-20	1073	1018	976	1056	934	983	959	987	1050	1011

Table 47: Numero di rilevamenti nella configurazione 46

	Configurazione 47									
Run 1-10	1002	1004	1015	972	990	956	987	1042	987	1036
Run 11-20	1081	1014	972	1041	932	948	937	981	1055	982

Table 48: Numero di rilevamenti nella configurazione 47

	Configurazione 48									
Run 1-10	991	996	1007	970	1006	953	990	1042	989	1011
Run 11-20	1073	1018	976	1056	934	983	959	987	1050	1011

Table 49: Numero di rilevamenti nella configurazione 48

B.1.2 Job Terminati

	Configurazione 1									
Run 1-10	1997	2024	1987	2106	1970	1930	2000	2000	1995	2058
Run 11-20	2043	1947	2001	1973	1935	1948	1950	1969	2013	1999

Table 50: Numero di rilevamenti nella configurazione 1

	Configurazione 2									
Run 1-10	1968	2026	2026	1982	1981	1973	1937	2032	2062	2022
Run 11-20	2011	1968	2014	1965	1946	1969	1921	1962	1951	2030

Table 51: Numero di rilevamenti nella configurazione 2

	Configurazione 3									
Run 1-10	1983	2008	1972	2096	1943	1929	1971	1995	1987	2026
Run 11-20	2034	1950	1986	1968	1922	1925	1962	1973	2007	1972

Table 52: Numero di rilevamenti nella configurazione 3

	Configurazione 4									
Run 1-10	1966	2026	2026	1982	1981	1973	1937	2032	2062	2022
Run 11-20	2011	1968	2013	1963	1943	1967	1921	1962	1951	2018

Table 53: Numero di rilevamenti nella configurazione 4

	Configurazione 5									
Run 1-10	1992	1989	1984	2062	1948	1922	1981	1992	1995	2030
Run 11-20	2025	1934	1976	1969	1924	1948	1943	1963	2010	1977

Table 54: Numero di rilevamenti nella configurazione 5

	Configurazione 6									
Run 1-10	1966	2026	2026	1982	1981	1973	1937	2032	2062	2022
Run 11-20	2011	1968	2014	1961	1946	1966	1921	1967	1950	2029

Table 55: Numero di rilevamenti nella configurazione 6

	Configurazione 7									
Run 1-10	1998	2012	1992	2106	1970	1930	1990	1991	1997	2032
Run 11-20	2042	1947	1997	1964	1937	1943	1963	1976	2003	2000

Table 56: Numero di rilevamenti nella configurazione 7

	Configurazione 8									
Run 1-10	1968	2026	2026	1982	1981	1973	1937	2032	2062	2022
Run 11-20	2011	1968	2014	1965	1946	1968	1921	1962	1951	2030

Table 57: Numero di rilevamenti nella configurazione 8

	Configurazione 9									
Run 1-10	1983	1992	1967	2097	1948	1929	1979	1993	1981	2007
Run 11-20	2036	1947	1975	1950	1930	1921	1951	1966	1994	1963

Table 58: Numero di rilevamenti nella configurazione 9

	Configurazione 10									
Run 1-10	1967	2026	2026	1982	1981	1973	1937	2032	2062	2021
Run 11-20	2011	1968	2013	1961	1946	1968	1921	1961	1951	2018

Table 59: Numero di rilevamenti nella configurazione 10

	Configurazione 11									
Run 1-10	1979	2006	1965	2074	1953	1927	1982	1998	2002	2024
Run 11-20	2028	1950	1959	1940	1930	1913	1945	1962	2004	1963

Table 60: Numero di rilevamenti nella configurazione 11

	Configurazione 12									
Run 1-10	1968	2026	2026	1982	1981	1973	1937	2032	2062	2020
Run 11-20	2011	1968	2015	1960	1946	1965	1921	1960	1948	2029

Table 61: Numero di rilevamenti nella configurazione 12

	Configurazione 13									
Run 1-10	1407	1385	1420	1386	1413	1337	1383	1428	1405	1432
Run 11-20	1428	1372	1415	1417	1354	1371	1357	1364	1415	1399

Table 62: Numero di rilevamenti nella configurazione 13

	Configurazione 14									
Run 1-10	1392	1386	1400	1401	1384	1326	1410	1452	1394	1416
Run 11-20	1495	1380	1415	1439	1338	1358	1330	1353	1410	1395

Table 63: Numero di rilevamenti nella configurazione 14

	Configurazione 15									
Run 1-10	1407	1385	1420	1385	1410	1336	1383	1425	1405	1430
Run 11-20	1428	1372	1415	1416	1354	1369	1356	1361	1409	1399

Table 64: Numero di rilevamenti nella configurazione 15

	Configurazione 16									
Run 1-10	1392	1386	1400	1401	1384	1326	1410	1452	1394	1416
Run 11-20	1495	1380	1415	1439	1338	1358	1330	1353	1410	1395

Table 65: Numero di rilevamenti nella configurazione 16

	Configurazione 17									
Run 1-10	1407	1385	1420	1382	1412	1337	1382	1421	1403	1432
Run 11-20	1425	1371	1415	1417	1354	1369	1353	1359	1410	1399

Table 66: Numero di rilevamenti nella configurazione 17

	Configurazione 18									
Run 1-10	1392	1386	1400	1401	1384	1326	1410	1452	1394	1416
Run 11-20	1495	1380	1415	1439	1338	1358	1330	1353	1410	1395

Table 67: Numero di rilevamenti nella configurazione 18

	Configurazione 19									
Run 1-10	1407	1385	1420	1386	1413	1337	1383	1425	1405	1432
Run 11-20	1428	1372	1415	1412	1354	1370	1357	1364	1408	1399

Table 68: Numero di rilevamenti nella configurazione 19

	Configurazione 20									
Run 1-10	1392	1386	1400	1401	1384	1326	1410	1452	1394	1416
Run 11-20	1495	1380	1415	1439	1338	1359	1330	1353	1410	1395

Table 69: Numero di rilevamenti nella configurazione 20

	Configurazione 21									
Run 1-10	1407	1385	1418	1385	1411	1336	1383	1423	1405	1430
Run 11-20	1428	1372	1415	1416	1354	1364	1356	1359	1412	1401

Table 70: Numero di rilevamenti nella configurazione 21

	Configurazione 22									
Run 1-10	1392	1386	1400	1401	1384	1326	1410	1452	1394	1416
Run 11-20	1495	1380	1415	1439	1338	1355	1330	1353	1408	1395

Table 71: Numero di rilevamenti nella configurazione 22

	Configurazione 23									
Run 1-10	1407	1385	1416	1382	1412	1337	1383	1421	1402	1426
Run 11-20	1425	1371	1415	1408	1354	1368	1357	1358	1401	1399

Table 72: Numero di rilevamenti nella configurazione 23

	Configurazione 24									
Run 1-10	1392	1386	1400	1401	1384	1326	1410	1452	1394	1416
Run 11-20	1495	1380	1415	1438	1338	1357	1330	1353	1409	1395

Table 73: Numero di rilevamenti nella configurazione 24

	Configurazione 25									
Run 1-10	1177	1204	1227	1195	1202	1132	1201	1243	1169	1227
Run 11-20	1259	1210	1203	1249	1163	1172	1129	1190	1232	1211

Table 74: Numero di rilevamenti nella configurazione 25

	Configurazione 26									
Run 1-10	1191	1196	1199	1195	1202	1137	1188	1243	1171	1213
Run 11-20	1280	1202	1183	1239	1137	1162	1112	1146	1242	1210

Table 75: Numero di rilevamenti nella configurazione 26

	Configurazione 27									
Run 1-10	1177	1204	1225	1195	1202	1132	1201	1238	1169	1227
Run 11-20	1259	1210	1203	1249	1163	1172	1129	1190	1231	1207

Table 76: Numero di rilevamenti nella configurazione 27

	Configurazione 28									
Run 1-10	1191	1196	1199	1194	1202	1137	1188	1243	1171	1213
Run 11-20	1280	1202	1183	1239	1137	1162	1112	1146	1242	1210

Table 77: Numero di rilevamenti nella configurazione 28

	Configurazione 29									
Run 1-10	1177	1203	1226	1194	1202	1132	1201	1238	1169	1227
Run 11-20	1259	1206	1203	1253	1163	1172	1129	1189	1232	1211

Table 78: Numero di rilevamenti nella configurazione 29

	Configurazione 30									
Run 1-10	1191	1196	1199	1194	1202	1137	1188	1243	1171	1213
Run 11-20	1280	1202	1183	1239	1137	1162	1112	1146	1242	1210

Table 79: Numero di rilevamenti nella configurazione 30

	Configurazione 31									
Run 1-10	1177	1204	1227	1195	1202	1132	1201	1241	1169	1227
Run 11-20	1259	1210	1203	1246	1163	1172	1129	1190	1231	1211

Table 80: Numero di rilevamenti nella configurazione 31

	Configurazione 32									
Run 1-10	1191	1196	1199	1195	1202	1137	1188	1243	1171	1213
Run 11-20	1280	1202	1183	1239	1137	1162	1112	1146	1242	1210

Table 81: Numero di rilevamenti nella configurazione 32

	Configurazione 33									
Run 1-10	1177	1204	1224	1195	1202	1132	1201	1236	1169	1226
Run 11-20	1259	1210	1203	1249	1163	1172	1129	1189	1225	1207

Table 82: Numero di rilevamenti nella configurazione 33

	Configurazione 34									
Run 1-10	1191	1196	1199	1194	1202	1137	1188	1243	1171	1213
Run 11-20	1280	1202	1183	1239	1137	1162	1112	1146	1242	1210

Table 83: Numero di rilevamenti nella configurazione 34

	Configurazione 35									
Run 1-10	1177	1203	1225	1194	1202	1132	1201	1238	1169	1228
Run 11-20	1259	1209	1202	1246	1163	1172	1129	1188	1235	1205

Table 84: Numero di rilevamenti nella configurazione 35

	Configurazione 36									
Run 1-10	1191	1196	1199	1194	1202	1137	1188	1243	1171	1213
Run 11-20	1280	1202	1183	1238	1137	1162	1112	1146	1242	1210

Table 85: Numero di rilevamenti nella configurazione 36

	Configurazione 37									
Run 1-10	1001	1004	1014	972	988	955	987	1042	987	1036
Run 11-20	1081	1013	968	1035	932	948	937	981	1048	982

Table 86: Numero di rilevamenti nella configurazione 37

	Configurazione 38									
Run 1-10	991	996	1005	970	1003	953	990	1042	989	1011
Run 11-20	1071	1018	976	1056	934	983	959	986	1050	1011

Table 87: Numero di rilevamenti nella configurazione 38

	Configurazione 39									
Run 1-10	1001	1004	1014	972	988	955	987	1041	987	1035
Run 11-20	1081	1013	968	1035	932	948	937	981	1047	982

Table 88: Numero di rilevamenti nella configurazione 39

	Configurazione 40									
Run 1-10	991	996	1005	970	1003	953	990	1042	989	1011
Run 11-20	1071	1018	976	1056	934	983	959	986	1050	1011

Table 89: Numero di rilevamenti nella configurazione 40

	Configurazione 41									
Run 1-10	1001	1004	1014	972	988	955	987	1039	987	1035
Run 11-20	1081	1012	968	1035	932	948	937	981	1048	982

Table 90: Numero di rilevamenti nella configurazione 41

	Configurazione 42									
Run 1-10	991	996	1005	970	1003	953	990	1042	989	1011
Run 11-20	1071	1018	976	1055	934	983	959	986	1050	1011

Table 91: Numero di rilevamenti nella configurazione 42

	Configurazione 43									
Run 1-10	1001	1004	1014	972	988	955	987	1042	987	1036
Run 11-20	1081	1013	968	1034	932	948	937	981	1047	982

Table 92: Numero di rilevamenti nella configurazione 43

	Configurazione 44									
Run 1-10	991	996	1005	970	1003	953	990	1042	989	1011
Run 11-20	1071	1018	976	1056	934	983	959	986	1050	1011

Table 93: Numero di rilevamenti nella configurazione 44

	Configurazione 45									
Run 1-10	1001	1004	1014	972	988	955	987	1040	987	1033
Run 11-20	1081	1013	968	1033	932	948	937	981	1044	982

Table 94: Numero di rilevamenti nella configurazione 45

	Configurazione 46									
Run 1-10	991	996	1005	970	1003	953	990	1042	989	1011
Run 11-20	1071	1018	976	1055	934	983	959	986	1050	1011

Table 95: Numero di rilevamenti nella configurazione 46

	Configurazione 47									
Run 1-10	1001	1004	1014	972	988	955	987	1039	987	1034
Run 11-20	1081	1012	968	1038	932	948	937	981	1051	982

Table 96: Numero di rilevamenti nella configurazione 47

	Configurazione 48									
Run 1-10	991	996	1005	970	1003	953	990	1042	989	1011
Run 11-20	1071	1018	976	1055	934	983	959	986	1050	1011

Table 97: Numero di rilevamenti nella configurazione 48