Progetto di Simulazione di Sistemi

Samuele Evangelisti

a.a. 2019/2020

Contents

List of Figures

A Codice Sorgente

A.1 Job

```
1  //
2  // This file is part of an OMNeT++/OMNEST simulation example.
3  //
4  // Copyright (C) 2006-2015 OpenSim Ltd.
5  //
6  // This file is distributed WITHOUT ANY WARRANTY. See the file
7  // 'license' for details on this and other legal matters.
8  //
9
10 #ifndef __QUEUEING_JOB_H
11 #define __QUEUEING_JOB_H
12
13 #include <vector>
14 #include "Job_m.h"
15
16 namespace queueing {
17
18 class JobList;
19
20 /**
21  * We extend the generated Job_Base class with support for split-join, as well
22  * as the ability to enumerate all jobs in the system.
23  *
24  * To support split-join, Jobs manage parent-child relationships. A
25  * relationship is created with the makeChildOf() or addChild() methods,
26  * and lives until the parent or the child Job is destroyed.
27  * It can be queried with the getParent() and getNumChildren()/getChild(k)
28  * methods.
29  *
30  * To support enumerating all jobs in the system, each Job automatically
31  * registers itself in a JobList module, if one exist in the model.
32  * (If there's no JobList module, no registration takes place.) If there
33  * are more than one JobList modules, the first one is chosen.
34  * JobList can also be explicitly specified in the Job constructor.
35  * The default JobList can be obtained with the JobList::getDefaultInstance()
36  * method. Then one can query JobList for the set of Jobs currently present.
37  */
38 class QUEUEING_API Job: public Job_Base
39 {
40     friend class JobList;
41     protected:
42         Job *parent;
43         std::vector<Job*> children;
44         JobList *jobList;
45         virtual void setParent(Job *parent); // only for addChild()
46         virtual void parentDeleted();
47         virtual void childDeleted(Job *child);
48         // progettoss
49         simtime_t absoluteDeadline;
50     public:
51         /**
52          * Creates a job with the given name, message kind, and jobList. If
53          * jobList==nullptr, the default one (or none if none exist) will be chosen.
54          */
55         Job(const char *name=nullptr, int kind=0, JobList *table=nullptr);
```

```
56
57          /** Copy constructor */
58          Job(const Job& job);
59
60          /** Destructor */
61          virtual ~Job();
62
63          /** Duplicates this job */
64          virtual Job *dup() const override {return new Job(*this);}
65
66          /** Assignment operator. Does not affect parent, children and jobList. */
67          Job& operator=(const Job& job);
68
69          /** @name Parent-child relationships */
70          //@{
71          /** Returns the parent job. Returns nullptr if there's no parent or it no longer
                exists. */
72          virtual Job *getParent();
73
74          /** Returns the number of children. Deleted children are automatically removed
                from this list. */
75          virtual int getNumChildren() const;
76
77          /** Returns the kth child. Throws an error if index is out of range. */
78          virtual Job *getChild(int k);
79
80          /** Marks the given job as the child of this one. */
81          void addChild(Job *child);
82
83          /** Same as addChild(), but has to be invoked on the child job */
84          virtual void makeChildOf(Job *parent);
85          //@}
86
87          /** Returns the JobList where this job has been registered. */
88          JobList *getContainingJobList() {return jobList;}
89
90          // progettoss
91          void setAbsoluteDeadline(simtime_t absoluteDeadline);
92
93  };
94
95  }; // namespace
96
97  #endif
```

Listing 1: "Job.h"

```
1  //
2  // This file is part of an OMNeT++/OMNEST simulation example.
3  //
4  // Copyright (C) 2006-2015 OpenSim Ltd.
5  //
6  // This file is distributed WITHOUT ANY WARRANTY. See the file
7  // 'license' for details on this and other legal matters.
8  //
9
10 #include <algorithm>
11 #include "Job.h"
12 #include "JobList.h"
```

```
13
14 namespace queueing {
15
16 Job::Job(const char *name, int kind, JobList *jobList) : Job_Base(name, kind)
17 {
18     parent = nullptr;
19     if (jobList == nullptr && JobList::getDefaultInstance() != nullptr)
20         jobList = JobList::getDefaultInstance();
21     this->jobList = jobList;
22     if (jobList != nullptr)
23         jobList->registerJob(this);
24 }
25
26 Job::Job(const Job& job)
27 {
28     setName(job.getName());
29     operator=(job);
30     parent = nullptr;
31     jobList = job.jobList;
32     if (jobList != nullptr)
33         jobList->registerJob(this);
34 }
35
36 Job::~Job()
37 {
38     if (parent)
39         parent->childDeleted(this);
40     for (int i = 0; i < (int)children.size(); i++)
41         children[i]->parentDeleted();
42     if (jobList != nullptr)
43         jobList->deregisterJob(this);
44 }
45
46 Job& Job::operator=(const Job& job)
47 {
48     if (this == &job)
49         return *this;
50     Job_Base::operator=(job);
51     // leave parent and jobList untouched
52     return *this;
53 }
54
55 Job *Job::getParent()
56 {
57     return parent;
58 }
59
60 void Job::setParent(Job *parent)
61 {
62     this->parent = parent;
63 }
64
65 int Job::getNumChildren() const
66 {
67     return children.size();
68 }
69
70 Job *Job::getChild(int k)
71 {
```

```
72      if (k < 0 || k >= (int)children.size())
73          throw cRuntimeError(this, "child index %d out of bounds", k);
74      return children[k];
75  }
76
77  void Job::makeChildOf(Job *parent)
78  {
79      parent->addChild(this);
80  }
81
82  void Job::addChild(Job *child)
83  {
84      child->setParent(this);
85      ASSERT(std::find(children.begin(), children.end(), child) == children.end());
86      children.push_back(child);
87  }
88
89  void Job::parentDeleted()
90  {
91      parent = nullptr;
92  }
93
94  void Job::childDeleted(Job *child)
95  {
96      std::vector<Job *>::iterator it = std::find(children.begin(), children.end(), child)
            ;
97      ASSERT(it != children.end());
98      children.erase(it);
99  }
100
101 void Job::setAbsoluteDeadline(simtime_t absoluteDeadline)
102 {
103     this->absoluteDeadline = absoluteDeadline;
104 }
105
106 }; // namespace
```

Listing 2: "Job.cc"

A.2 Source

```
1   //
2   // This file is part of an OMNeT++/OMNEST simulation example.
3   //
4   // Copyright (C) 2006-2015 OpenSim Ltd.
5   //
6   // This file is distributed WITHOUT ANY WARRANTY. See the file
7   // 'license' for details on this and other legal matters.
8   //
9
10  package org.omnetpp.queueing;
11
12  //
13  // A module that generates jobs. One can specify the number of jobs to be generated,
14  // the starting and ending time, and interval between generating jobs.
15  // Job generation stops when the number of jobs or the end time has been reached,
16  // whichever occurs first. The name, type and priority of jobs can be set as well.
17  // 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"

```
//
// 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_SOURCE_H
#define __QUEUEING_SOURCE_H

#include "QueueingDefs.h"

namespace queueing {

class Job;

/**
 * Abstract base class for job generator modules
 */
class QUEUEING_API SourceBase : public cSimpleModule
{
    protected:
        int jobCounter;
        std::string jobName;
        simsignal_t createdSignal;
    protected:
        virtual void initialize() override;
        virtual Job *createJob();
```

```
31          virtual void finish() override;
32  };
33
34
35  /**
36   * Generates jobs; see NED file for more info.
37   */
38  class QUEUEING_API Source : public SourceBase
39  {
40      private:
41          simtime_t startTime;
42          simtime_t stopTime;
43          int numJobs;
44
45      protected:
46          virtual void initialize() override;
47          virtual void handleMessage(cMessage *msg) override;
48  };
49
50
51  /**
52   * Generates jobs; see NED file for more info.
53   */
54  class QUEUEING_API SourceOnce : public SourceBase
55  {
56      protected:
57          virtual void initialize() override;
58          virtual void handleMessage(cMessage *msg) override;
59  };
60
61  }; //namespace
62
63  #endif
```

Listing 4: "Source.h"

```
1   //
2   // This file is part of an OMNeT++/OMNEST simulation example.
3   //
4   // Copyright (C) 2006-2015 OpenSim Ltd.
5   //
6   // This file is distributed WITHOUT ANY WARRANTY. See the file
7   // 'license' for details on this and other legal matters.
8   //
9
10  #include "Source.h"
11  #include "Job.h"
12
13  namespace queueing {
14
15  void SourceBase::initialize()
16  {
17      createdSignal = registerSignal("created");
18      jobCounter = 0;
19      WATCH(jobCounter);
20      jobName = par("jobName").stringValue();
21      if (jobName == "")
22          jobName = getName();
23  }
```

```
24
25 Job *SourceBase::createJob()
26 {
27     char buf[80];
28     sprintf(buf, "%.60s-%d", jobName.c_str(), ++jobCounter);
29     Job *job = new Job(buf);
30     job->setKind(par("jobType"));
31     job->setPriority(par("jobPriority"));
32     job->setAbsoluteDeadline(simTime() + par("jobRelativeDeadline"));
33     return job;
34 }
35
36 void SourceBase::finish()
37 {
38     emit(createdSignal, jobCounter);
39 }
40
41 //---
42
43 Define_Module(Source);
44
45 void Source::initialize()
46 {
47     SourceBase::initialize();
48     startTime = par("startTime");
49     stopTime = par("stopTime");
50     numJobs = par("numJobs");
51
52     // schedule the first message timer for start time
53     scheduleAt(startTime, new cMessage("newJobTimer"));
54 }
55
56 void Source::handleMessage(cMessage *msg)
57 {
58     ASSERT(msg->isSelfMessage());
59
60     if ((numJobs < 0 || numJobs > jobCounter) && (stopTime < 0 || stopTime > simTime()))
        {
61         // reschedule the timer for the next message
62         scheduleAt(simTime() + par("interArrivalTime").doubleValue(), msg);
63
64         Job *job = createJob();
65         send(job, "out");
66     }
67     else {
68         // finished
69         delete msg;
70     }
71 }
72
73 //---
74
75 Define_Module(SourceOnce);
76
77 void SourceOnce::initialize()
78 {
79     SourceBase::initialize();
80     simtime_t time = par("time");
81     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 5: "Source.cc"

A.3 Router

```
//
// 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 6: "Router.ned"

```
//
// This file is part of an OMNeT++/OMNEST simulation example.
//
```

```
4  // Copyright (C) 2006-2015 OpenSim Ltd.
5  //
6  // This file is distributed WITHOUT ANY WARRANTY. See the file
7  // 'license' for details on this and other legal matters.
8  //
9
10 #ifndef _QUEUEING_ROUTER_H
11 #define _QUEUEING_ROUTER_H
12
13 #include "QueueingDefs.h"
14
15 namespace queueing {
16
17 // routing algorithms
18 enum {
19       ALG_RANDOM,
20       ALG_ROUND_ROBIN,
21       ALG_MIN_QUEUE_LENGTH,
22       ALG_MIN_DELAY,
23       ALG_MIN_SERVICE_TIME,
24       // progettoss
25       ALG_PSSRANDOM
26 };
27
28 /**
29  * Sends the messages to different outputs depending on a set algorithm.
30  * Sends the messages to first queueNumber-th queues.
31  */
32 class QUEUEING_API Router : public cSimpleModule
33 {
34     private:
35         int routingAlgorithm;  // the algorithm we are using for routing
36         int rrCounter;         // msgCounter for round robin routing
37         // progettoss
38         int queueNumber;
39     protected:
40         virtual void initialize() override;
41         virtual void handleMessage(cMessage *msg) override;
42 };
43
44 }; //namespace
45
46 #endif
```

Listing 7: "Router.h"

```
1  //
2  // This file is part of an OMNeT++/OMNEST simulation example.
3  //
4  // Copyright (C) 2006-2015 OpenSim Ltd.
5  //
6  // This file is distributed WITHOUT ANY WARRANTY. See the file
7  // 'license' for details on this and other legal matters.
8  //
9
10 #include "Router.h"
11
12 namespace queueing {
13
```

```
14 Define_Module(Router);
15
16 void Router::initialize()
17 {
18     const char *algName = par("routingAlgorithm");
19     if (strcmp(algName, "random") == 0) {
20         routingAlgorithm = ALG_RANDOM;
21     }
22     else if (strcmp(algName, "roundRobin") == 0) {
23         routingAlgorithm = ALG_ROUND_ROBIN;
24     }
25     else if (strcmp(algName, "minQueueLength") == 0) {
26         routingAlgorithm = ALG_MIN_QUEUE_LENGTH;
27     }
28     else if (strcmp(algName, "minDelay") == 0) {
29         routingAlgorithm = ALG_MIN_DELAY;
30     }
31     else if (strcmp(algName, "minServiceTime") == 0) {
32         routingAlgorithm = ALG_MIN_SERVICE_TIME;
33     }
34     else if (strcmp(algName, "pssRandom") == 0) {
35         routingAlgorithm = ALG_PSSRANDOM;
36     }
37     rrCounter = 0;
38     int qn = par("queueNumber").intValue() - 1;
39     if (qn < 0 || qn > gateSize("out") - 1)
40         throw cRuntimeError("Invalid queue number");
41     else
42         queueNumber = qn;
43 }
44
45 void Router::handleMessage(cMessage *msg)
46 {
47     int outGateIndex = -1;  // by default we drop the message
48
49     switch (routingAlgorithm) {
50         case ALG_RANDOM:
51             outGateIndex = par("randomGateIndex");
52             break;
53
54         case ALG_ROUND_ROBIN:
55             outGateIndex = rrCounter;
56             rrCounter = (rrCounter + 1) % gateSize("out");
57             break;
58
59         case ALG_MIN_QUEUE_LENGTH:
60             // TODO implementation missing
61             outGateIndex = -1;
62             break;
63
64         case ALG_MIN_DELAY:
65             // TODO implementation missing
66             outGateIndex = -1;
67             break;
68
69         case ALG_MIN_SERVICE_TIME:
70             // TODO implementation missing
71             outGateIndex = -1;
72             break;
```

```
73
74          case ALG_PSSRANDOM:
75              outGateIndex = intuniform(0, queueNumber);
76              break;
77
78          default:
79              outGateIndex = -1;
80              break;
81      }
82
83      // send out if the index is legal
84      if (outGateIndex < 0 || outGateIndex >= gateSize("out"))
85          throw cRuntimeError("Invalid output gate selected during routing");
86
87      send(msg, "out", outGateIndex);
88  }
89
90  }; //namespace
```

Listing 8: "Router.cc"