《通信网理论与技术基础》教材配套实验

爱尔兰混合制系统

实验指导书

1 实验目标

利用 OMNeT++软件，构建爱尔兰混合制系统，并对呼叫等待概率和呼损概率进行数值分析验证。

2 实验背景

爱尔兰混合制系统，用肯德尔记号表示为 M/M/s/n/∞。它是指输入为一个泊松流，有 s 条中继线，服务时间满足指数分布，系统容量 n 有限的一类排队系统。系统容量 n 的值等于中继线的条数 s 和等待队列中可存储呼叫数之和。当新的呼叫到达时，若有中继线空闲，则呼叫立即随机占用任何一条空闲的中继线，并完成接续通话；若所有中继线都繁忙，但系统中已经存在的呼叫数（在中继线处接受服务的与正在等待的呼叫）小于 n 时，则呼叫进入等待队列；否则呼叫被拒绝。等待队列的排队规则是先入先出（FIFO），只要有中继线空闲，等待队列中队首的呼叫先去中继线上接受服务。

3 实验方法提示

3.1 理论值计算

(1) 呼叫等待概率

呼叫等待概率是指在呼叫抵达系统时，所有中继线都繁忙，但同时等待队列中有可用空间这一特定情形发生的概率。

业务量的强度通常称为呼叫量，理论值为：

$$a = \frac{\lambda}{\mu} \qquad \text{（式 1）}$$

其中λ代表呼叫的到达率、μ代表单条中继线的服务率。

由状态转移图以及概率的归一性得，系统空闲的概率为：

$$p_0 = \frac{1}{\sum_{k=0}^{s-1}\frac{a^k}{k!}+\frac{a^s}{s!}\cdot\frac{s}{s-a}[1-(\frac{a}{s})^{n-s+1}]} \quad \text{（式 2）}$$

其中 a 代表呼叫量，s 代表中继线的条数，n 代表系统的容量。

因此呼叫等待的概率为：

$$C_n(s,a) = \sum_{k=s}^{n} p_k = \frac{a^s}{s!}\frac{s}{s-a}[1-(\frac{a}{s})^{n-s+1}]p_0 \quad \text{（式 3）}$$

(2) 呼损概率p_n

呼损概率是指在呼叫抵达系统时，所有中继线都繁忙，并且等待队列已排满，呼叫被拒绝这一特定情形发生的概率。其理论值为：

$$p_n = B_n(s,a) = \frac{a^n}{s!s^{n-s}}p_0 \quad \text{（式 4）}$$

其中 a 代表呼叫量，s 代表中继线的条数，n 代表系统的容量，p_0代表系统空闲的概率。

3.2 仿真

(1) 仿真图

在 OMNeT++软件中，编写简单模块 Source、Exchange、Server 和 Sink 建立仿真。其中，Source 模块负责产生到达率为 lambda 的呼叫；Exchange 模块在中继线有空闲时负责将呼叫送至空闲的中继线、在中继线都繁忙但等待队列未满时将呼叫存储至等待队列、当中继线都繁忙且等待队列已满时拒绝呼叫；Server 模块负责模拟中继线，并按照服务率 mu 服务呼叫；Sink 模块负责模拟呼叫结束。在仿真中，不同中继线的服务率相同。

仿真图如下：

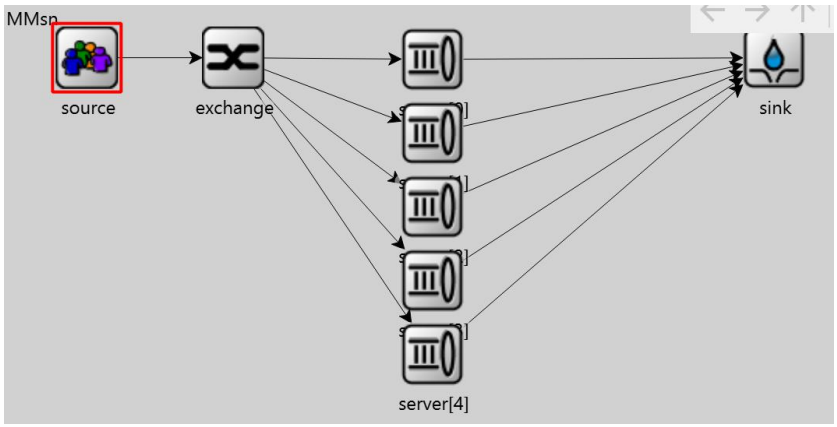

图 1 “爱尔兰混合制系统”的排队模型的仿真图

(2) 参数设置

在表 1 中，参数“lambda”代表 Source 模块中泊松流的到达率；参数“queueCapacity”代表等待队列可容纳的呼叫数上限；参数“mu”代表 Server 模块服务呼叫的服务率；参数“numServers”代表服务台的个数，该参数不在简单模块中，而是在用来定义复合模块的“MMsn.ned”文件中。

表 1 “M/M/m/m”排队模型的仿真参数设置

网络模块	参数名
Source	lambda
Exchange	queueCapacity
Server	mu
——	numServers

(3) 代码提示——容器类 cQueue

创建呼叫的方法，参见“实验 3”代码提示部分的“cMessage 类”；在 Qtenv 仿真中自定义中继线条数和在 Exchange 模块中获取中继线忙闲状态的方法，参见“实验 4”的代码提示部分。

容器类 cQueue 在 OMNeT++中可以表示排列，可以存储 cObject 派生的类对象，比如 cMessage、cPar 等。该容器类一般默认遵守先入先出（FIFO）的原则。

- 声明容器类

在.h 文件中用如下代码声明了一个名为 queue 的 cQueue 类型的容器类。

```
cQueue queue;
```

- insert()函数

insert()函数用于将元素存储到 cQueue 容器类中。

在下面的代码中利用 cMessage 类构建了一个对象，并将其存储到名为 queue 的 cQueue 容器类中。

```
// 创建新的呼叫
cMessage *job = new cMessage("job");
// 将呼叫储存在队列中
queue.insert(msg);
```

- pop()函数

pop()函数用于将元素从 cQueue 容器类中删除。

在下面的代码中，取出了队列 queue 中队首的元素，并赋值给了 msg 指针。

```
// 从队列中取出一个呼叫
```

```
cMessage *msg = check_and_cast<cMessage *>(queue.pop());
```

- getLength()函数

getLength()函数用于获取并返回 cQueue 容器中存储的元素的个数。其用法如下：

```
queue.getLength();
```

4 实验步骤

1）编写 Source、Exchange、Server 和 Sink 四个简单模块。

2）利用编写的四个简单模块构建复合模块。

3）设置参数的值，进行仿真。

4）仿真结束后，利用仿真得到的数据对系统进行分析。

5 提交要求

学生需要提交一份实验报告，并将实验报告、源代码、数据打包成压缩包上交。其中实验报告内容包括但不限于：

1）研究背景与问题描述

2）总体思路与方案论证

3）结果分析与结论

4）课程学习收获与体会建议