王宇晨 王禾雨

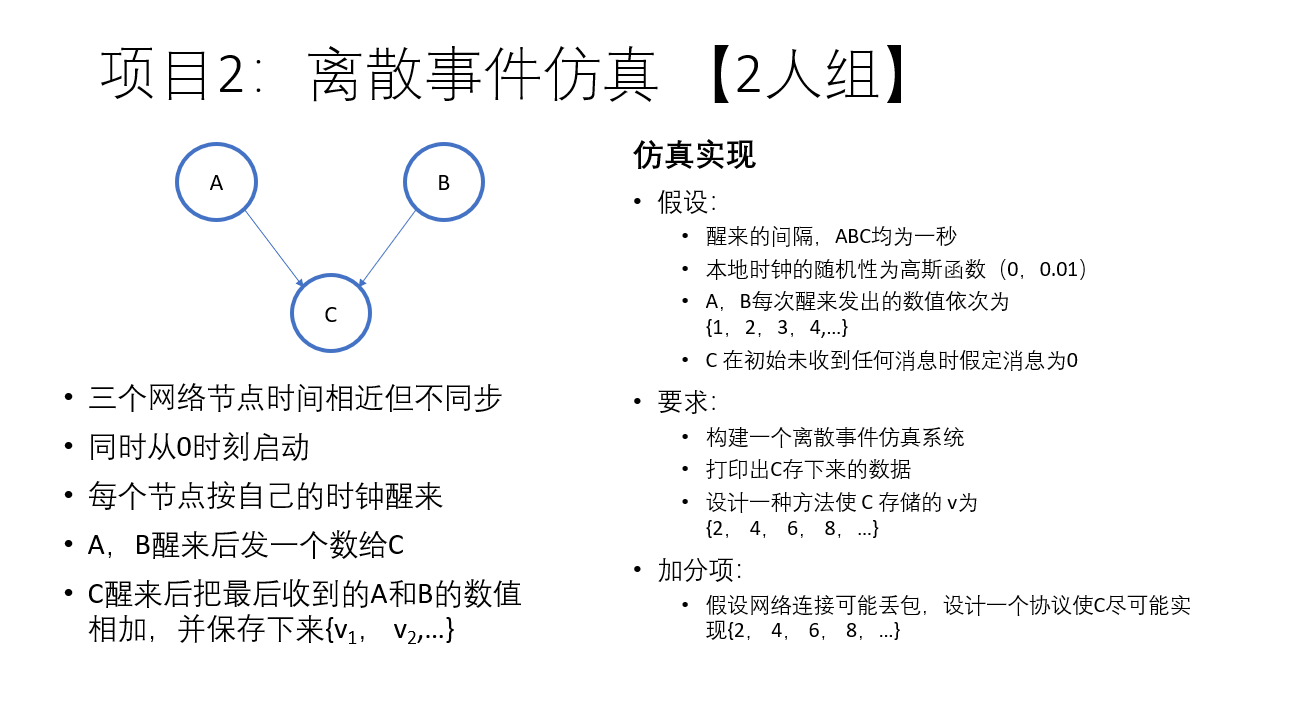
刘劼

CS33852

2020/7/29

项目报告： Project 2 离散事件仿真

1. 项目任务描述



1. 项目进程日志 (Change Log)

Part 1：使用SimPy进行仿真：（中途停止）

1. 7/17 SimPy\_Crude\_Version 诞生，包含仿真的基本框架及最基础模型的仿真模拟演示。（Created by 王禾雨）
2. 7/18 SimPy\_v2 诞生，经过逻辑修改使得仿真模型能够实现A、B、C节点自由醒来并传输/接收数据。（Corrected by 王宇晨）
3. 7/19-7/21 王禾雨不断修正模型，基本达成模拟仿真要求。
4. 7/22 项目期中汇报，SimPy\_Version被叫停。

Part 2：不借助工具进行仿真：

1. 7/23 Work\_Without\_SimPy\_Version 启动
2. 7/25 Work\_Without\_SimPy\_Framework 建立，能够实现abc的依次唤醒（数据传输方式还未确定） （Created by 王宇晨）
3. 7/28 王禾雨对模型进行修正，添加数据传输的方法，并包含数据传输丢包情况下的仿真。（Bug反馈：在某些情况下丢包数据并不能被相应method进行修正）
4. 7/29 王宇晨对“丢包数据修正”的相关method进行逻辑再修改，基本上解决丢包数据修正的问题。
5. 7/29-7/30 王禾雨对代码进行逻辑修正，使模型更符合真实传输情况；王宇晨完成项目答辩报告。
6. 项目理解：
7. 事件流程：网络节点A、B、C同时从0时刻启动，启动后按照各自的本地时钟醒来。A、B、C的本地时钟相近（均为1s），但不同步（误差的随机性服从Gauss(0,0.01)）。A、B为数据发送节点，醒来后将数据发送给C；C为数据接收节点，醒来后接受A、B发送的最近的一次数据，并把他们相加并储存。
8. 仿真中的技术问题：
9. 设计方法使得A、B、C在启动后能依次被唤醒，并完成传输/接收
10. 设计Sender A/B输出数据的产生方式（用什么方法产生1，2，3，4……）
11. 这种数据产生方式是否方便进行修正以使得C接收的数据为（2，4，6，8……）
12. 数据在丢包后的解决方法（数据产生、传输方式相关）
13. 为解决目标问题需要弄清楚的模型要点：
14. 节点唤醒的顺序：由于每一个节点启动后到下一次唤醒之间，每一次的time\_delay满足以下数学关系：time\_delay = 1 + gauss(0,0.01) (s)。所以说对于一个节点，例如A，可能第一次唤醒经历的time\_delay是0.997s，而下一次的time\_delay却是1.012s。所以，由于节点唤醒的随机性的存在，A、B、C的唤醒顺序可能不断交替。
15. 数据输出的分析：数据输出节点A、B在自然状况下的输出信息为{1，2，3，4……}。若A、B、C交替唤醒，此时C中存储的数据才有可能为{2，4，6，8……}。但由“节点唤醒的顺序”已知，这种情况并不能维持，所以说，要使得C的存储数据符合要求，就需要设定一定的协议来完成。
16. 丢包问题的分析：如果发生了丢包，导致的直接影响是A、B的数据传输不能及时到达C。可能导致的间接影响是，C由于没有及时收到更新后的A、B的传输值，导致加和了之前最近一次收到的A、B传输值，从而使得C存储的数据不满足要求的序列顺序。
17. 模型构建
18. 解决问题的思路：

分步实现模型构建：

* 1. 首先完成A、B、C的启动，并且保证可以分别唤醒。这是整个模型构建的基础。

针对A、B这样的数据输出节点，我们创建了Sender类，并且定义了awake、sleep等method来实现A、B在唤醒后所要执行的发送数据的操作（通过awake这一method）。接着在发送了数据后，再次休眠直到经过又一个time\_delay。

针对C这样的数据接收节点，我们创建了Receiver类，并且除了定义awake、sleep这些method外，还定义了add\_data这一方法。因为C除了接收数据，还要进行对接收到的A、B数据进行加和。这样，一个最基础的模型的framework就构建完成了

* 1. 接着我们小组设计了A、B节点的数据传输方法，和C在接收到A、B数据后的处理方法。

我们小组设计的使A、B完成数据正确传输，并使得C对数据正确存储的方式是通过创建“Packet”这个类来规定传输的数据需要包含那些信息（包含的信息有：数据本身、序列ID、数据传输来源），然后Sender A\B产生这些信息，接着通过Receiver类下的put这一method来获得A、B送出的“Packet”信息。

这样设计的理由是，由先前分析的“仿真模型的要点”可知，仅由单一数据而没有任何供系统确认的特征元素是很难实现数据有顺序的输出\接收的。

而数据带上了序列ID这一可供系统确认的特征元素就可以较为简单的实现输出\接收的有序进行了。

* 1. 接下来我将解释在这种数据传输方式下，且丢包的情况时有发生时，模型是怎么工作的。

首先Sender A/B开始运行，开始工作，并产生带有序列ID的数据，这些数据通过put函数输送给Receiver C进行接收。

在Receiver C中会提前产生一个所有元素均为-1的列表用于接收数据。其中-1并没有明确意义，但可作为检测特征值来检测在列表的这个位置是否有收到A/B发送的数据。这就为丢包的检测提供了方法。

在这个过程中另一项可以依赖的数据就是序列ID。在Receiver C中存储的序列ID会随最新接收到的数据不断更新，它告诉了系统应该检查列表中的多少个数据来验证丢包。

接着在验证丢包之后，Receiver C会把采集到的信息传回Sender A/B，接着Sender A/B会再次尝试发送相应信息给Receiver C。如果这次发送的信息也丢包了，那么在下一次醒来时，系统会再次以同样方式处理这些问题。

1. 总结/心得

通过这次对离散事件模型的建模和仿真，我和我的队友都在这个过程中成长了不少。我们都是第一次接触离散事件仿真，但经过不断学习、查阅资料，我们也顺利的完成了模型的构建。在建模的过程中，我们感受到了用现成建模工具SimPy来仿真离散事件给建模带来的便利，也亲身实践了在不借助工具的条件下进行仿真的过程。我们对于模型的理解也随着建模的逐渐完成而逐渐加深，在最后，我们也感受到了这个简单的数据传输模型背后可能引出的广阔的拓展面。经过这几周的学习，随着Project的完成，我们构建模型的能力和对于模型本身的理解已经得到了很大的提升。