1. 附录场景D测试数据的定义数据中,Core1应该为无线节点,如何知道是无线节点,参数如何定义,结合TC3-7中,Topo.xml文件中“<EndSystems>”字段表示有无线建模。无线拓扑的定义,得与有线交换机得有区别,有特定的字段来定义表示无线.



如上图所示，每个平台的无线网络接入点由红色部分表示，无线传输速率由绿色部分表示。

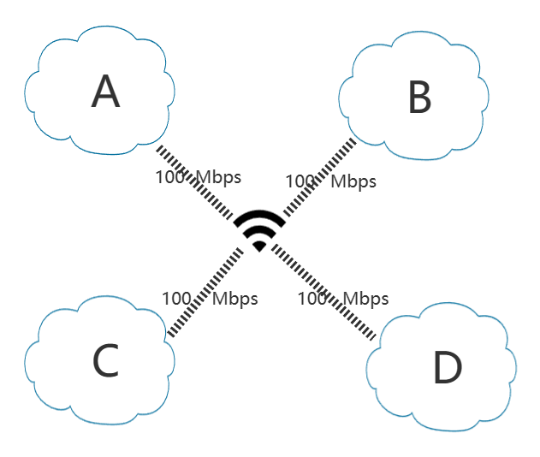
以图为例，平台1的网络命名为“A”，该网络的无线接入点为“core1”。平台1内部的两个主机host1与host2通过core1接入无线网络，其无线传输速率为100Mbps（绿色部分定义了Network-A-Speed=100）。

而<Switches>和<Links>部分定义的speed=10和BW=10是在以太网中使用的，表示平台内部以太网速率为10Gbps。这是因为“无线网络场景”和“混合以太网与无线网场景”可以共用一个拓扑文件。

“无线网络场景”使用<EndSystems>里定义的无线传输参数。

“混合以太网与无线网场景”在平台内部通信时，使用<Switches>和<Links>里定义的以太网参数，跨平台通信时使用<EndSystems>里定义的无线传输参数。

无线拓扑如下图所示：



平台接入点CoreX连接到无线网络WirelessNet中，这部分工作无需在Topo.xml文件里显示指明。仿真软件会根据Core、Network和EndSystems-Speed自动在后台将CoreX以设定好的无线传输速率连接到WirelessNet。

1. 截屏数据打印后,是否清晰可见.

好的，我们会注意检查图片

1. 逐步将测试用例补充完整,例如:”K8S格式的YAML文件导入功能测试”

好的，会和其他用例一样配上图示

1. 格式排版整齐,如7.2.2.2,文字左对齐.
2. TC2-2任务群总完成时间的含义是什么,本处HEFT是否考虑了DAG,如果没有则表示容器之间无关联性,则总完成时间是表示什么,是否更多的考虑资源分配的负载和链路负载的均匀性,使用的模块更少等因素,供参考.容器分配 7 - 2 - 6中stage-in含义是啥
3. 本处HEFT有考虑DAG,输入的任务群有形成一个简单的DAG关联。任务群总完成时间是这一系列容器全部完成的时间。若无关联性，则容器会同时运行，总完成时间是所有的容器的最大完成时间
4. 之后会补充对比
5. Stage-in表示这个容器是调度模块为了生成任务关系而自己产生的假容器，不占用资源，只是代码需要，实际上不应该在后端显示，已经删除，之后会更新图片
6. 图 7 - 2 – 8和图 7 - 2 – 9, HEFT调度相对于K8S算法,在在集群的平均CPU利用率和平均ram利用率都下降,原因是什么.

得到这个结果和测试用例有一定关系。在容器数少时HEFT算法会倾向于保证一台物理机同一时间只运行一个容器。这个用例下K8s调度的结果是多个容器拥挤在一台物理机上，而HEFT的结果是每一台物理机同时只运行一个容器，所以资源的利用率下降。但是如果是容器数多的情况下，可能得不到这样的结果。

1. 容器调度给出料7种算法,各种算法有什么优缺点,分别适合什么应用,用户如何选择

之后会补充说明

1. TC2-8中, 测试场景的测试结果是什么

之前搭建的K8s集群已经取消了所以没配上结果图，更据之前K8s对比测试的结果，在TC2-8用例的测试场景下是吻合的。

1. 图7 - 3 - 27 将任务建模为消息,Appinfo中每一条消息都有周期,为什么在这个地方定义发送3条消息. 仿真可以设定总仿真时间.

在这个测试用例中，我们设置了一个任务，该任务包含一条周期性消息。

图7 - 3 – 27的意思是“一个消息周期性发三遍”。

可以看到在图中表格里，只有一行内容，所以是一条消息。但该行有参数count=3，period=0.02，所以这条消息会周期性地发三遍，发送间隔时间为0.02s。

1. TC3-7中缺少TDMA机制的测试

补充TDMA测试：

（8）无线TDMA测试

测试场景：

场景C。平台4个，每个平台内两个主机节点。主机跨平台通信均以无线接入方式。

任务负载使用简单任务。主机A、B同时跨平台向主机C发送消息。

|  |  |  |  |
| --- | --- | --- | --- |
| 测试编号 | #TC3-8 | 测试名称 | 无线TDMA测试 |
| 测试目的 | 测试软件支持TDMA机制。 | | |
| 前置条件 | 调度模块完成生成容器，并提供仿真输入。  仿真器Simuator.exe已启动。  操作面板已启动。 | | |
| 测试步骤 | 操作内容 | | 操作图示 |
| 1 | 按照“TC1-1—HEFT算法调度功能测试”的执行步骤，在操作面板上点击“调度容器”后，自动将“Host.xml、AppInfo.xml和容器调度结果Assign.json文件” 上传至仿真系统。 | |  |
| 2 | 在操作面板上点击“上传Topo.xml”，将网络拓扑文件上传至仿真系统。 | | 图1 |
| 3 | 在操作面板上点击“开始仿真”。 | | 图2 |
| 4 | 在Simulator.exe的OutputFiles目录下查看仿真延迟结果，输出文件output\_latency.xml。 | | 图3、图4 |
| 合格判据 | | | |
| 在无线网络场景下，正确输出延迟文件output\_latency.xml。两条消息的延迟相差0.0001s（即一个TDMA时间片）。 | | | |

操作图示：



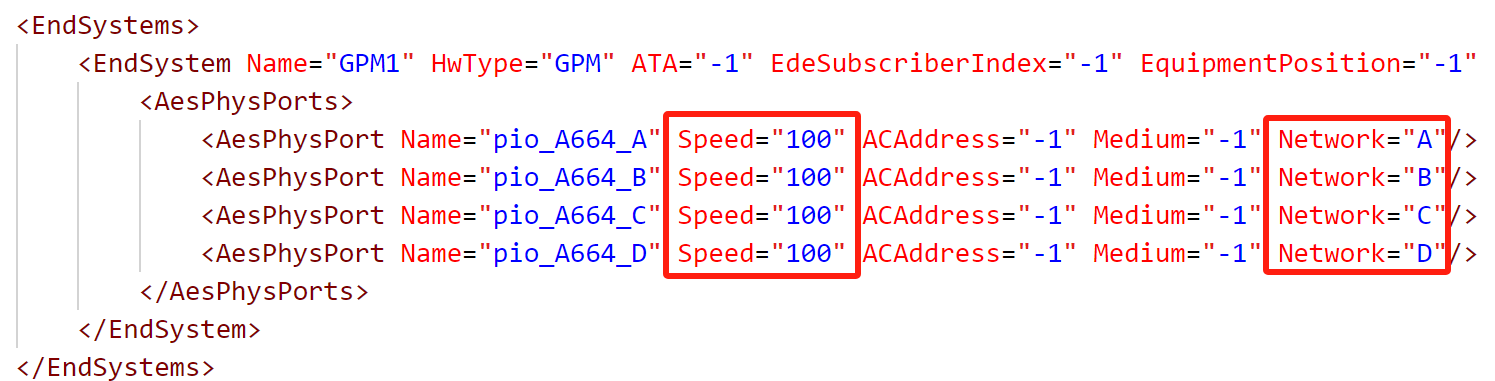
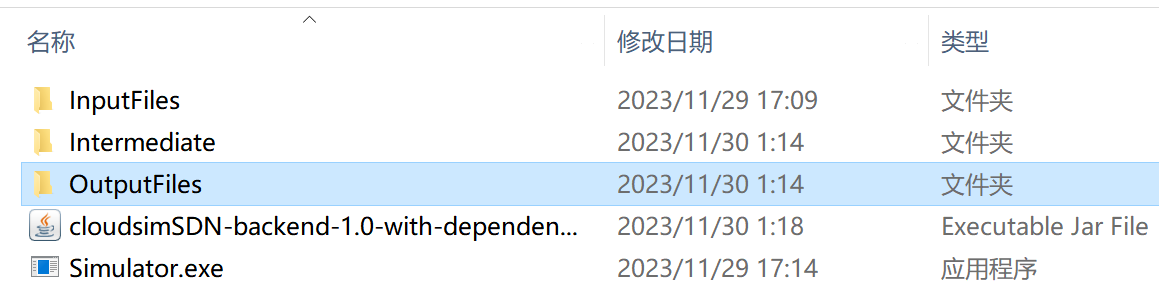


图1 无线设置

四个平台A、B、C、D的无线传输速率均设为100Mbps。



图2 运行仿真



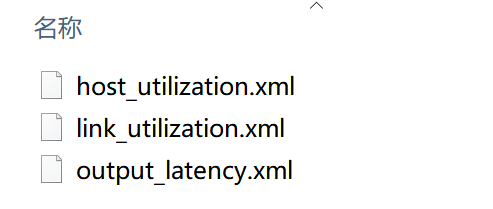


图3 仿真输出

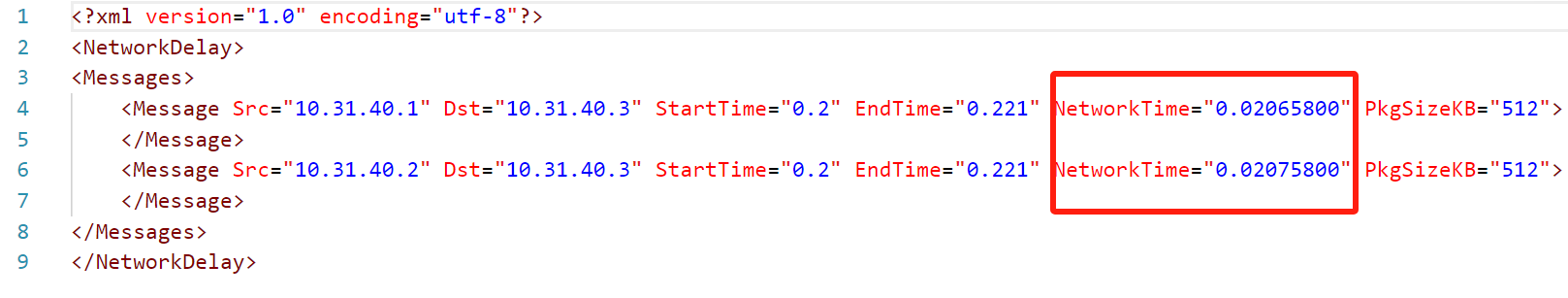


图4 仿真时延

output\_latency.xml输出了两条消息的仿真时延。

由于主机A和主机B都跨平台向主机C发送消息。两条消息在TDMA的调度队列里，每次发送一个时间片的数据。最终，A最后一个时间片的数据先发送完毕，B紧随其后，二者相差一个时间片0.0001s的时延。

仿真结果正确，证明软件支持TDMA机制。

1. TC3-10中,DAG测试,应该测的是,相同DAG定义的任务,在不同调度算法(不考虑DAG任务调度算法,咱们考虑DAG的调度算法HEFT)情况下通过仿真,仿真出完成任务的执行时间,任务负载,链路负载,需要的平台规模都不同吧. 以体现出咱们提出的算法优越性,不知理解是否正确.

这样测试确实更能体现DAG的特性，之后会修改用例

1. 容器调度仿真中有容器”等待、暂停和恢复”,TC3-11是容器调度仿真过程中吗,好像写的是资源调度过程的内容

TC3-11修改为：

（11）容器暂停功能测试

测试场景：

场景A。选择3个主机节点的小型以太网。

任务负载使用简单任务。容器A在0.5s到1s内运行，每隔0.1s周期性发送消息。

|  |  |  |  |
| --- | --- | --- | --- |
| 测试编号 | #TC3-11 | 测试名称 | 容器暂停功能测试 |
| 测试目的 | 测试软件能正确将容器在指定的时间暂停，并在指定的时间后恢复运行 | | |
| 前置条件 | 调度模块的exe可执行文件，前端界面exe可执行文件。  Host3.xml、AppInfo8.xml、ContainerInfo8.xml、Topo.xml输入文件 | | |
| 测试步骤 | 操作内容 | | 操作图示 |
| 1 | 双击点击“调度模块.exe”，运行后端 | |  |
| 2 | 双击打开“操作界面.exe” | |  |
| 3 | 在下拉框点击选择“HEFT” | |  |
| 4 | 点击“上传Host.xml”按钮，选择Host3.xml输入文件 | |  |
| 5 | 点击“上传AppInfo.xml”按钮,选择AppInfo8.xml输入文件 | |  |
| 6 | 点击“上传Container.xml”按钮，选择ContainerInfo8.xml输入文件 | |  |
| 7 | 点击“调度容器” | |  |
| 8 | 在操作面板上点击“上传Topo.xml”，将网络拓扑文件上传至仿真系统。 | | 图1 |
| 9 | 在操作面板上点击“开始仿真”。 | | 图2 |
| 10 | 在OutputFiles目录下查看仿真延迟结果，输出文件output\_latency.xml。 | | 图3 |
| 11 | 容器暂停部分，在操作面板上容器填1（AppInfp.xml中第一个容器），暂停开始时间填0.15（开始运行0.15秒后暂停），结束时间填0.2（暂停持续0.2秒）。 | | 图4 |
| 12 | 点击“调度容器” | |  |
| 13 | 在操作面板上点击“上传Topo.xml”，将网络拓扑文件上传至仿真系统。 | |  |
| 14 | 在操作面板上点击“开始仿真”。 | |  |
| 15 | 在OutputFiles目录下查看仿真延迟结果，输出文件output\_latency.xml。 | | 图5 |
| 合格判据 | | | |
| 对比图3和图5。容器暂停时间内不再发送消息，恢复运行后继续发送消息。 | | | |

操作图示：



图1 上传拓扑文件



图2 运行仿真

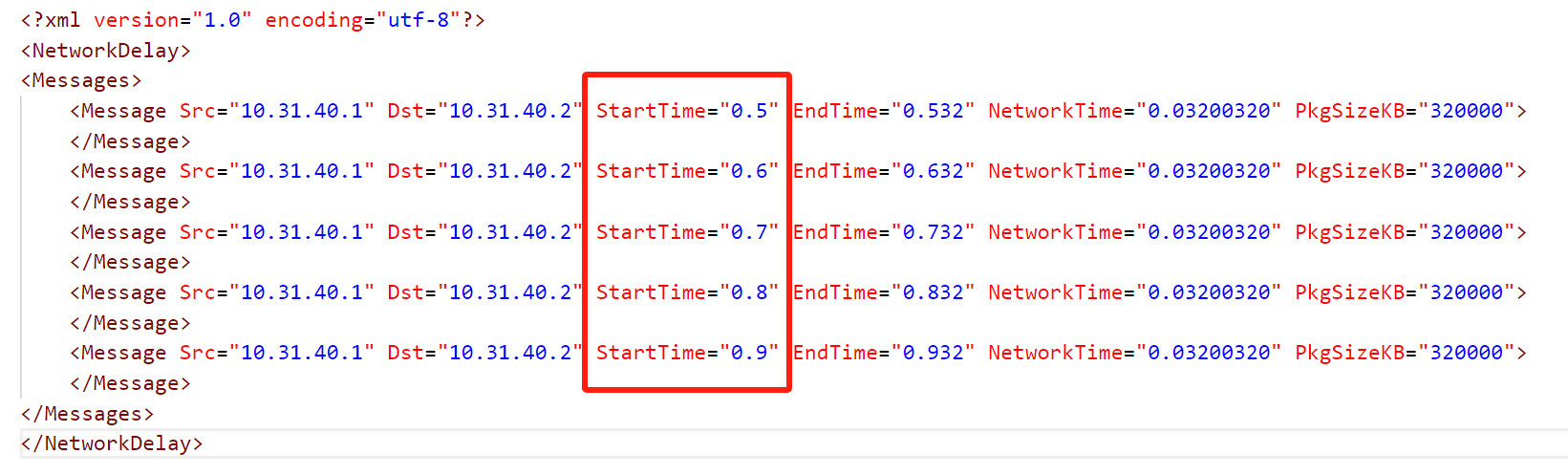


图3 不暂停的仿真延迟

容器A每隔0.1s发送消息。



图4 设置容器暂停

容器A暂停开始时间设为启动后0.15s，持续时间设为0.2s。

即0.65s至0.85s内容器暂停。

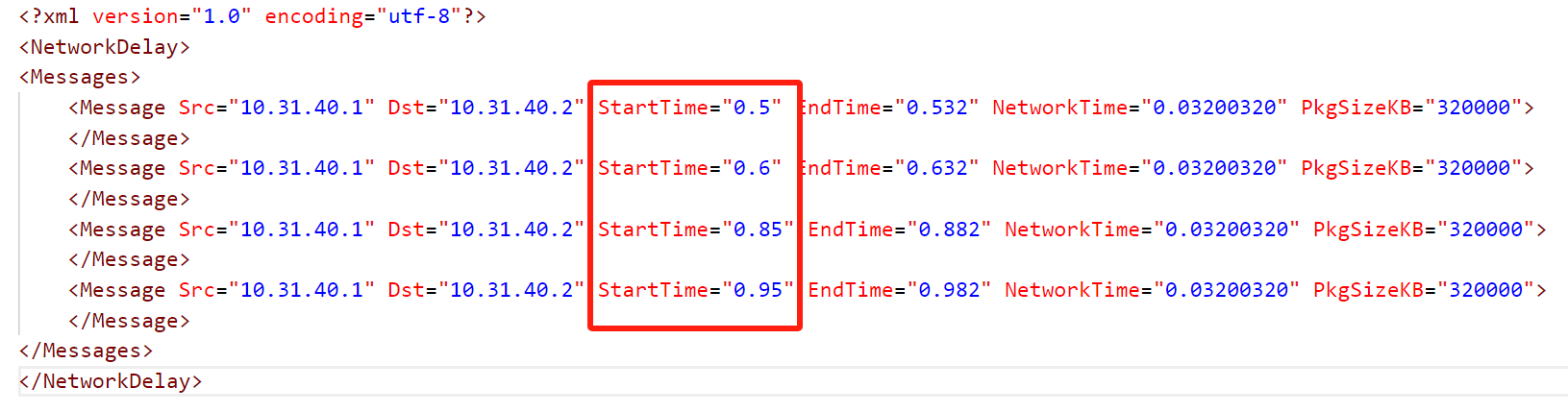


图5 暂停后的仿真延迟

容器在0.65s后不再发送消息，直到0.85s恢复后，继续发送消息。

软件能正确将容器在指定的时间暂停，并在指定的时间后恢复运行。

1. 技术协议书部分内容解读:
2. 状态显示：
3. 支持系统运行后故障显示、容器生成及资源分配方案显示、计算及通信仿真状态和结果显示。
4. 支持出错位置能自动定位到架构模型数据文件(在Appinfo与其他输入文件信息不一致的情况下,能报错,不是输入专门的故障文件)。
5. 能显示每一次分配结果的当前状态，如评分值。(评分值如何定义,参上述问题7)
6. 重构控制：(针对仿真过程中,输入特定故障后仿真过程如何变化,如重新分配等)
7. 支持用户故障注入。
8. 支持故障概率分布建模
9. 支持故障修复策略定义
10. 支持故障监视
11. 容器分配方案控制:用户能可视化调整容器的分配方案
12. 故障记录：校验警告或者错误记录在LOG文件中，需要包括出错或者警告的位置信息。