元组空间副本机制的设计与实现

叶如锐 周韬 叶宇翔 陈晨阳

1. 研究课题



Adhoc移动网络中，LIME中每一Agent具有本地元组空间，LIME提供的元组空间融合技术，可以使不同Agent之间实现对元组空间的互访问，但当某一Agent退出该网络时，其携带的元组空间信息在该网络中消失。

元组空间复制技术，是使每个Agent本地的元组空间，具有曾与之共享的元组空间的信息，这样就可以实现Agent在不同网络中的信息共享。

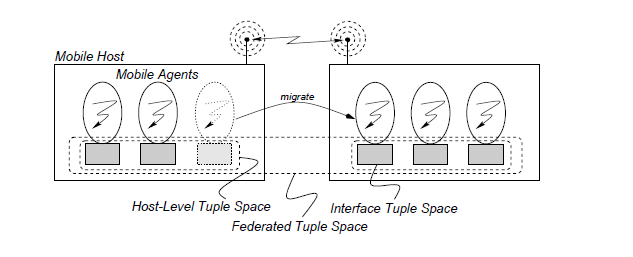
1. 实现目标

1） 为应用编写Agent提供API接口

2） 对应用屏蔽不同host的元组空间复制的机制。

3） 为应用屏蔽元组的更新机制

1. 实现背景LIME



LIME即Linda in a Mobile Environment，LIME定义了可广泛用于展示物理或逻辑机动性的应用的协调层，它借用并适应了Linda的通信模型。

·Tuple

* \_fields ------> Vector
* \_formal ------> Vector

Formal

Template:

Hello.class

String.class

hello

asdf

Field:

you

hello

addActual()

在Linda中，进程通过tuple space进行通信，通信的参数叫做Tuple，它包括了两种类型actual和formal，分别代表数据本身和数据类型。

·Tuple Space

LIME Tuple Space是一个基于LIME的应用的最基本的组件，它和存在的其他LIME Tuple Space相互作用，为应用提供了一系列的操作函数，包括Shared Control，Out，in以及Reaction，分别用于共享Tuple，将Tuple写入Tuple Space，将Tuple从Tuple Space中读出，以及对Tuple Space变化的响应。

·Reaction

在动态的移动环境中，对Tuple Space的变化的响应必然十分重要，所以LIME扩展了基本Linda Tuple Space的reaction观念。当某个匹配的Tuple在Tuple Space中被找到，则LIME会进行一些预先设定好的操作。我们可以为当前或者是目的地位置的所有template设置reaction。

1. 设计实现

·Replication

ReplicableTuple

LimeTuple

Data

VrsionedTuple

将原本Tuple Space扩展两层，一层为Versioned LIME Tuple Space，第二层为Replicable LIME Tuple Space。每一层通过调整和委派底层功能，以实现用户可见的操作。通过在这两层中添加一些属性和操作，使得原来的LIME能够支持Tuple的快速复制。

·Tuple Extend

<data> → <origCur , origDest , isReplica, TupleID,Vrsion,userdata>

在实际实现过程中，为了方便，把两层合并为一层，并在原有的limetuple内存入实现复制所需的5个Field：

origCur: 最初放入元组的AgentLocation

origDest: 元组最初的目的地AgentLocation

isReplica: 元组是否是复制版本

TupleID: 元组的ID，此ID由ReplicationTupleSpace管理，而不是系统的tupleID

Version: 元组的版本号

·Actions

当在移动环境中一个Agent加入到一个网络中，则所有Agent触发reaction，所有Agent共享Tuple Space，在这个过程中，根据各个Tuple Space中的Tuple的属性的不同，Tuple有着不同的操作，有的替换有的不替换，严格根据其属性相应地复制Tuple。

|  |  |  |
| --- | --- | --- |
| Replication Mode | MASTER  ANY | 只从原始的Agent复制  从任何副本中复制 |
| Consistency Mode | NEVER  MASTER  ANY | 不更新  只从原始Agent更新  从任何副本中更新 |

1. 代码实现

·ReplicableTuple

**public** **ReplicableTuple**(String name , **int** id){

tuple = **new** Tuple();

tuple.addActual(AgentLocation.UNSPECIFIED); // origCur

tuple.addActual(AgentLocation.UNSPECIFIED); // origDest

tuple.addActual(IS\_MASTER); // isReplica

tuple.addActual(**new** LimeTupleID(id)); // ID

tuple.addActual(DEFAULT\_VERSION); // Version

}

一个ReplicableTuple的构造函数,其中name为ReplicableTupleSpace的name，id为由ReplicableTupleSpace管理的id。

·Reaction

**public** RegisteredReaction **addReplicaRequest**(

ReplicableTuple template,

**int** replicationMode,

**int** consistencyMode) {

**try** {

Reaction reaction =

**new** UbiquitousReaction(template.getTuple(),

**new** ReplicationListener(replicationMode, consistencyMode,template),

Reaction.ONCEPERTUPLE);

**return** lts.addWeakReaction(**new** Reaction[]{reaction})[0];

} **catch** (TupleSpaceEngineException e) {

e.printStackTrace();

}

**return** **null**;

}

用户通过调用addReplicaRequest函数来实现元组的复制，其中，template指定要复制的元组，两个mode分别为复制的模式。

1. 遇到的问题以及解决方案

**·问题一：**元组的系统ID无法从外面获得。

解决方案：由于要实现Update功能，所以如果无法正确地获得ID，将无法判断此元组是否是可以更新的。同时，如果由LimeTupleSpace管理ID，则无法让插入的Tuple更新而不改变ID。基于此，由ReplicatableTupleSpace提供一套ID的管理机制会更容易是满足我们的要求。

**·问题一：** 由于更新了template导致只有一次复制

解决方案：在Lime中，Tuple没有提供一个clone函数来实现元组的赋值。所以直接使用等号赋值会导致原本的template被改变，从而不能再次满足其匹配条件。通过重新new出一个ReplicableTuple，然后把每一个field都复制过去，解决了此问题。