文章编号: 1001-2486(2002) 01-0085-04

基于 Agent 的数据传输管理系统*

李 琦, 陈国强, 华祖跃
(国防科技大学机电工程与自动化学院, 湖南 长沙 410073)

摘要: 为了提高分布交互仿真中数据传输、管理的效率, 结合软件 Agent 技术, 提出了基于 Agent 的分布仿真数据管理及传输的结构, 并将其应用于分布式仿真平台网络接口部件的研制开发中, 形成智能网络接口。智能网络接口的应用可以使用户更加有效地管理分布的、异构的集成环境并提高各仿真站点的仿真速度。

关键词: 仿真; 智能网络接口部件; 数据管理; Agent
中图分类号: TP309 文献标识码: A

Data Management System Based on Multi-Agent

LI qi, CHEN Gue-qiang, HUA Zu-yue
(College of Mechatronics Engineering and Automation, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: In order to improve the efficiency of the data transfer and management in DIS, the framework based on soft Agent of the distributed simulation data management and transfer is presented . The soft Agent technicality has been used in the development of the network interface unit to form the intelligent network interface unit used in the distributed simulation platform. It is easier and more effective to manage the distributed, heterogeneous and integrated environment as well as to promote the simulation speed in every simulation station.

Key words: simulation; the intelligent network interface unit; data management; Agent

在分布仿真中, 为了减少网上各仿真站点传输 PDU 的压力, 各仿真站点都设置了对仿真实体进行推算、平滑, 然后对 PDU 数据打包上网等一系列操作。另外, 仿真对象的物理位置及姿态表示是一个基本问题, 各仿真站点都要进行相应的坐标转换, 因为不同的仿真站点可能采用不同的坐标系来表示, 如地心坐标系, 大地坐标系及局部坐标系来表示对象的物理位置, 对象的姿态也有欧拉角、四元数和方向余弦等不同的表示方法。总之, 为了提高各仿真站点的实时仿真速度, 我们设计了实时网络接口部件, 其中, 引入了 Agent 技术, 形成智能网络接口部件, 用以加强实时网络接口部件间的数据管理, 实现部件间的灵活、开放、可扩充的协调和管理, 也使网络接口部件的效率得以充分的体现。

1 将软件 Agent 引入分布仿真数据管理系统

基于网络环境下的分布交互仿真其主要特点有: 交互(INTERACTIVE), 并发(CONCURRENT), 分布(DISTRIBUTED), 主动对象(ACTIVE OBJECTS), 移动对象(MOBILE OBJECTS), 协同工作(COOPERATION), 具有内部属性状态, 具有生命周期(从创建到消亡), 其中最主要的是交互、并发、分布。

Agent 是具有自主能力的实体抽象, 具有自主性、协调性等特点。

因此, 我们认为 Agent 概念及技术适应于分布仿真的要求, Agent 尤其是 Multi-Agent 被引入到分布仿真中更有利于分布交互仿真的设计与实现。

分布仿真中引入 Agent, 在计算机实现时如何表示 Agent 呢? 我们认为 Agent 是在一定环境中连续自主运行的实体, 能感知与之相关的环境变化并产生相应的行动。因此, Agent 的形式化定义为:

* 收稿日期: 2001-10-16
基金项目: 国家部委资助项目(4.4.3.7)
作者简介: 李琦(1964—), 女, 副教授, 博士生。

Agent 是一个五元组描述的实体, 即 Agent= 〈A, I, S, T, K〉,

其中: A 表示 Agent 主体, 即 Agent 名称或标识, 应具有唯一性;

I——用户环境, 指用户界面和通信接口;

S——状态集, 是描述 Agent 内部状态的集合, Agent 的行为实际上是由一个状态转换到另一个状态的过程;

T——功能集, 定义的功能或行为;

K——知识源, 描述 Agent 行为所需要的知识、数据、推理规则以及有关资源, 体现为数据结构、数据库、知识库等。

2 基于 Agent 的智能网络接口的关键技术

Agent 是一个抽象实体, 它是自治的, 可以对自身环境、操作环境和环境变化采取相应的行动。

Agent 结构: Agent 的内部结构一般包括功能模块、知识库、实用函数库、外推算算法库、通信接口库和网络接口库, 如图 1 所示。

(1) 功能模块

代理的功能实现模块, 可以是一个子 Agent, 也可以包含其它功能和 Agent。

(2) 通信和网络接口

用来对交互信息进行封装, 同时对其它 Agent 发来的信息进行解析。

(3) 黑板

外界信息内容和各功能单元工作过程的中间信息存放区, 用于和外界以及内部各功能单元之间交换数据。

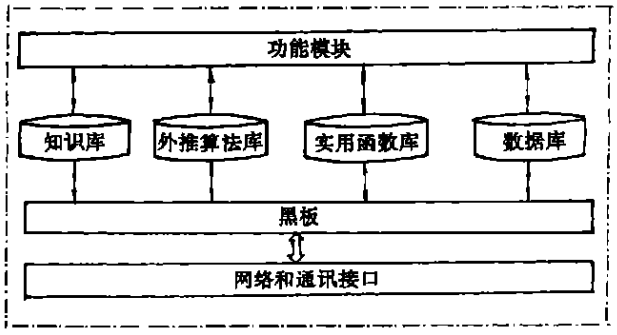

图 1 Agent 的内部结构

Fig 1 Internal framework of Agent

(4) 知识库

存放与本 Agent 有关的本地知识以及规则。

(5) 数据库

存放与本 Agent 有关的本地数据。

(6) 实用函数库

主要由以下几类实用函数组成:

① 连接/ 断开网络接口部件;

② 帧调度: 初始化并启动帧调度过程;

③ 计时函数;

④ 查询函数, 例如:

查询实体的类型: int LookUp_ EntityType(char * entity_ name , Entity_ type * type)

查询实体的名字: char LookUp_ Entity_ Name (Entity_ type * type)

……等等。

⑤ 创建本地实体, 例如:

增加一个本地定义的实体到实体表中: Entity_ Table * AddEntity (entity_ state_ pdu * pdu , int id, char * type , char * alt_ type , int force_ id , char * marking , int DR_ aly , long appearance)

填写一个实体状态: Entity_ Table * FillEntity (entity_ state_ pdu * pdu , int id, char * type , char * alt_ type , int force_ id , char * marking , int DR_ aly , long appearance)

⑥ 更新本地实体

Entity_ Table * ModifyEntity(Entity_ state_ pdu * pdu)

⑦ PDU 响应函数

void AddcallBack(int type , void (* func) Genetic_ pdu * , int priority)

实用库函数是开放的, 用户可以根据自己的需要增加新的库函数。

(7) 模型推算算法库

模型推算主要采用一阶、二阶外推公式进行推算。这些推算公式中的变量 X 是泛指, 可代表三个位置变量 X, Y, Z 和三个欧拉角变量中的任意一个, 其中任意一个变量的偏差超过预定的阈值都将引起状态更新。

算法库中的外推算法有一阶、二阶、三阶方程。例如, 二阶外推有:

①$X_i = X_0 + V_0 ih + \alpha_0 (ih)^2/2$

α_0 是最近一次状态更新的加速度;

②$X_i = X_0 + V_0\left[ih + \frac{(ih)^2}{2\Delta t_{-1}}\right] - V_{-1}\frac{(ih)^2}{2\Delta t_{-1}}$

V_{-1}是对应 t_{-1}时刻的状态更新的速度值, 用$(V_0 - V_{-1})/\Delta t_{-1}$来逼近 α_0

③$X_i = X_0\left[1 - \frac{(ih)^2}{2\Delta t_{-1}^2} + V_0\left[ih + \frac{(ih)^2}{2\Delta t_{-1}^2}\right] + X_{-1}\frac{(ih)^2}{2\Delta t_{-1}^2}\right.$

④$X_i = X_0\left[1 + \frac{ih}{\Delta t_{-1}} + \frac{(ih)^2}{2\Delta t_{-1}^2}\right] - X_{-1}\left[\frac{ih}{\Delta t_{-1}} + \frac{(ih)^2}{2\Delta t_{-1}^2} + \frac{(ih)^2}{2\Delta t_{-1}\Delta t_{-2}}\right] + X_{-2}\frac{(ih)^2}{2\Delta t_{-1}\Delta t_{-2}}$这里 $\Delta t_{-2} = t_{-1} - t_{-2}$, t_{-1}表示在 t_0之前的状态更新时间, t_{-2}表示在 t_{-1}之前的状态更新时间, V_{-2}则是对应 t_{-2}时刻的状态变量。

从仿真的结果分析可知, 在一般情况下, 阶数较高的算法比阶数较低的算法逼近程度会好一些。在阈值较大的情况下, 高阶算法的优势不是很明显。用户可根据自己的仿真需要选择相应的外推公式, 通过预定义文件将其选择传递给 Agent, 另外, 在仿真过程中可进行自动转换, 在缺省的情况下选择二阶外推公式。

在用以上算法进行模型推算作模型选取时, 我们还提供了可利用自适应模型转换法进行模型选取的方法。对于仿真系统中初始出现的目标, 要在较短的时间周期内确定其状态模型, 并跟踪其状态模型的变换, 才能实现多个仿真系统间的实时协同交互, 因此, 我们利用如下方法进行模型选取:

对于实体 E, 它在时刻 t 的模型概率 $\mu(t)$定义为:

$\mu(t) = [\mu_1(t) \ldots \mu_n(t)]^T$

其中, n 为与运动相关的模型总数, $\mu_i(t)$表示 t 时刻模型i 的概率, $0 \leqslant \mu_i(t) \leqslant 1$, $\sum_{i=1}^{n}\mu_i(t) = 1$, 当 $\mu_k(t) = \max[\mu_i(t)]$时, 可以选取模型 k 为当前应用的DR 模型。

对 n 类模型而言, 从模型 i 转换成模型j 的先验概率称为模型转换率, 用 P_{ij}表示, n 类模型间的转换可用模型转换概率矩阵 $\boldsymbol{P}$ 表示:

根据概率模型和模型转换概率, 可以推导出下一时刻实体 E 的模型概率为:

$\mu(t) = \frac{1}{c}\boldsymbol{L}_N\boldsymbol{P}^T\mu(t-1)$

$\boldsymbol{L}_N = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) = \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ \vdots & & \ddots \\ 0 & \cdots & \lambda_n \end{bmatrix}$　$c = \lambda^T\boldsymbol{P}^T\mu(t-1)$　　$\lambda^T = [\lambda_1 \lambda_2 \ldots \lambda_n]^T$

λ是由实体E 按模型i 计算的预测残差的似然函数, c 为归一化因子。

初始出现的实体模型可直接由先验知识确定。

3　智能网络接口部件的功能及流程

智能网络接口部件与应用程序间关系如图 2 所示。

网络接口部件的基本功能为:

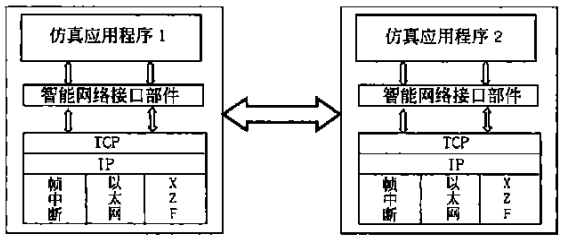

图 2 智能网络接口部件与应用程序间关系

Fig 2 The relationship between the intelligent network interface unit program

完成帧同步锁定及协调; 利用监控 Agent 和 过滤 Agent 对本站点所需的 PDU 进行适当处理, 发送进入本站点的数据包; 对于本站点的仿真实体进行推算, 对于超过阈值的 PDU 打包上网; 对于远程结点的 PDU 进行推算, 超过阈值的 PDU 送本站点仿真机; 轮询网络接口; 模型推算; 坐标转换; 网络通信是整个系统各部分进行交换的核心, 使每个分散的仿真站点成为一个有机的大系统, 提供各个节点数据之间的交互能力。通信部分不关心数据的具体含义, 仅为应用提供一个透明的接口而只注重数据的流向以及各个数据要求任务之间的协调。

网络接口部件相当于服务器端 server, 仿真机相当于客户端 client 。仿真机只与本机的接口部件之间存在通信关系。它的所有请求都由接口部件处理, 接口部件间的通信采用 TCP/ IP 协议, 使用 Winsocket 编程。接口部件内部采用共享内存的方式通信。每个接口部件运行一个通信主线程, 处理实际的网络通信。当应用程序有请求时, 主线程对这个请求注册登记后, 开启一个辅线程进行相应的处理, 并将实时数据和一些过程数据存放在共享内存区。

智能网络接口部件的软件结构如图 3 所示。

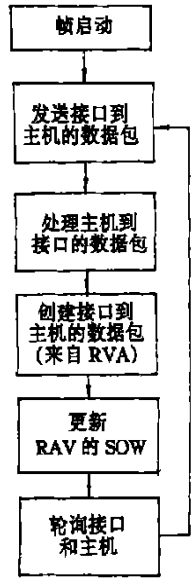

图 3 软件流程

Fig 3 Flowchart

4 结束语

分布式仿真中的数据管理是一个庞大复杂的系统, 如何对它进行有效而正确的管理已成为一个重要的课题。本文为此专门设计了网络接口部件并将可移动的软件 Agent 技术引入其中, 有效地提高了仿真数据管理的效率, 减轻了各站点仿真计算的负担。移动 Agent 技术对于未来分布式系统的设计、实现都有着重要的意义。这两者的结合为开发一个灵活、可伸缩、高效的分布式仿真系统提供了更为广阔的前景。

参考文献:

[1] 李琦, 华祖跃. 智能网络接口部件设计[R]. 长沙国防科学技术大学, 1997.

[2] 刘弘等. 软件 Agent 的构筑[J]. 计算机科学. 1998.

[3] Konolige K, Pollack M E. A representationalist theory of intention [J]. In: Bajcsy R, ed. Proc of the 13th Int. Joint Conf. on Artificial Intelligence. California: Morgan Kaufman Pulishers, 1999.