第33卷 第16期 计 算 机 工 程 2007年8月
Vol.33 No.16 Computer Engineering August 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)16—0264—03 文献标识码：A 中图分类号：TP391

基于 Agent 的人机对话系统的设计与实现

徐凯华，张德干，姚 琳
(北京科技大学信息工程学院计算机系，北京 100083)

摘 要：在人机对话领域中，多任务和人性化要求的提高，导致人机对话软件的规模日益庞大和结构日渐复杂，该文运用 Agent 技术提出一种人机对话系统框架，有效地实现了系统多任务分解和信息共享，并基于该框架实现了一个英文学习的人机对话系统。
关键词：多 Agent 系统；人机对话；中心转发；端到端通信

Design and Realization of Human-computer Dialog System Based on Agent

XU Kai-hua, ZHANG De-gan, YAO Lin
(Department of Computer, School of Information Engineering, University of Science and Technology Beijing, Beijing 100083)

【Abstract】 In the field of the human-computer dialog more tasks to be furnished and humanistic requests about computer result that the size of software about the human-computer dialog is more greatness and that the structure is more complex, so the software engineers are called for working hard. In that condition, the human-computer dialog system frame based on agent is studied. It is effective for decomposing many tasks and sharing information of the system. The human-computer dialog of English study based on this frame is designed.
【Key words】 multi-agent system; human-computer dialog; center transmit; end-to-end communication

人机对话系统一直是人工智能领域内的研究热点，随着语音技术的日渐成熟，对话管理逐渐被认为是对话系统的关键问题，是整个系统的核心功能体现。由于种种限制，目前的人机对话系统大多是面向单个任务领域内的对话，而且只能在单机上运行，很少考虑对话过程涉及的多主题、主题切换、主题间的信息共享，以及对系统的复杂功能进行任务分解，使分解后的各个功能模块能运行在不同终端上通过通信合作实现更加强大的功能，使得系统易于扩展。

Agent 技术是解决这些问题的最好方法，利用 Agent 技术可以很容易地实现任务的分解，多 Agent 之间又能通过一些通信机制实现 Agent 之间的信息共享。

1 Agent 技术

Agent 技术是近几年分布式人工智能(DAI)等领域研究的热点之一，并且已经在计算机科学的许多领域得到了广泛的应用。Agent 是一个具有自治性、自适应性、协同性和智能性的内部驱动的软件实体，它能作用于自身和环境，并能对环境作出适应性的反应。按其应用又可以分为单 Agent 系统和多 Agent 系统。

单 Agent 系统主要用于实现本地的任务。多 Agent 系统是由一组独立的但又协同工作的 Agent 构成的，Agent 是其基本的组成单元，又是独立运行的实体。在多 Agent 系统中，各 Agent 相互协商和协作，以完成某一共同任务，其中每个 Agent 可以根据负载变化和其他 Agent 的情况来协调自身的行为，对实现目标和资源的使用作合理的安排和调整以避免冲突。

2 系统设计

结合 Agent 技术，本文提出一种新型的人机对话英文学习多 Agent 系统。

2.1 系统结构

系统模型主要由 3 大类实体构成：Agent，Container 以及 DS，其结构如图 1 所示。Agent 是系统中软件模块的最基本的封装。一个 Agent 可以被认为是一个封装了行为以及功能的能够自主运行并有一定运行目的的模块。Agent 本身会携带自己的接口描述以及相关的知识表示。Container 是每个计算设备所需要驻留一个守护进程(daemon process)，它为同一计算设备上的 Agent 提供一个本地的运行环境和相关的系统服务。这些服务包括系统管理服务以及文件共享以及获取服务。

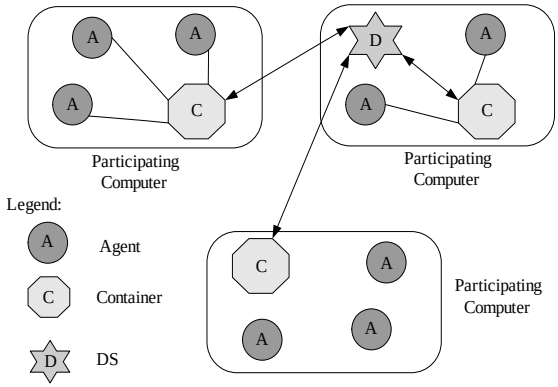

图 1 系统结构

DS 是在一个运行环境中全局唯一的系统服务进程，每个

基金项目：国家自然科学基金资助项目(60503024)
作者简介：徐凯华(1979－)，女，硕士研究生，主研方向：人工智能及网络；张德干，博士；姚 琳，副教授
收稿日期：2006-11-05 **E-mail：**xkh1027@126.com

计算设备上的 Container 需要和 DS 建立连接，从而组成一个全局的运行环境。DS 可以驻留在任何一台计算机上，它利用自己维护的 Agent 的运行状态，Agent 的相互关联信息以及全局的 Agent 的接口描述以及知识表示来为 Agent 提供注册、消息转发、资源管理和运行时的环境管理等基础服务。注册服务是 DS 提供的最基本的服务。在这里消息转发和资源管理服务分别由 DS 的子模块消息转发模块以及资源管理模块提供。从图 1 可以看出系统中所有的 Agent 都必须与本地主机上的 Container 连接，而 Container 直接与 DS 连接，Agent 和 DS 都只能够直接与 Container 进行通信，Agent 和 DS 之间的任何对话都必须以 Container 为中介。

2.2 通信模型

“中心转发”和“端到端”通信模式相结合如图 2 所示。

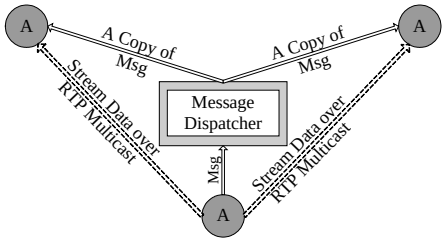

图 2 混合通信模式

(1)中心转发通信模式

该模式是基于“发布-订阅”的通信模式。在这种模式中，Agent 之间所有的消息都通过 DS 转发，很容易实现“一对多”的通信而且要维护的连接数较少。然而这样 DS 将成为通信的瓶颈，而且可能导致任意两条连接的传输延迟。所以当两个 Agent 之间通信较频繁时，需要另一种通信模式——端到端的通信。

(2)端到端通信模式

该模式是一种在通信双方之间直接建立物理连接的通信方式，一般采用 TCP/IP 协议，具有端到端的确认，可以保证信息的安全到达。虽然端到端通信具有安全可靠的特性，但在多 Agent 系统中采用这种方式存在这样的问题：当系统中 Agent 数量较大时，由于每一对 Agent 之间都存在一对连接，因此通信量是巨大的，往往使系统不堪重负。为克服这种缺陷，充分利用它的安全可靠性，在本方案的设计中，采用了有限的端到端方式，只对实时性和可靠性要求高的消息采用端到端方式，既限制了通信量又实现了端到端的通信。

(3)通信实现方案

1)中心转发的实现

这一功能主要是由 DS 的子模块消息转发模块完成的。消息转发模块的最底层分别是输入消息和事件的处理者以及输出消息的包装者。在它们之上分布是 3 个功能模块消息控制模块，转发模块以及消息缓存模块。控制模块用于处理相关的控制信息，并修改数据库(用来存放消息和 Agent 的注册信息)中的消息定义信息以及状态信息。Agent 之间传递的消息会被发送到转发模块，转发模块会根据数据库中存储的发送者和接收者的关联关系将消息转发到对应的模块，当发送者和接收者之间的关联关系因为资源的调度而无法确定消息应该被发送到谁时，消息转发模块会把这些消息暂存在消息缓冲模块中，等消息控制模块收到相应的控制信息并能确定消息最终的目标，再从消息缓冲区中取出消息并发送到相应的对象。

2)端到端通信的实现

①监听端口

在使用 TCP 连接实现端到端的通信时，每个 Agent 都要有一个监听地址。每个 Agent 的地址由两部分构成：IP 地址和监听端口号。每个 Agent 的监听端口号仅由本地 Container 分配即可。具体步骤如下：

Agent 与 Container 建立连接，并将注册请求发送给 Container，Container 为该 Agent 选择一个监听端口，并将监听端口号和本地主机的 IP 地址加入 Agent 的注册消息中，发送给 DS；

Agent 注册成功后，要使用 QueryListenPort 原语向自己的 Container 查询自己的监听端口，并在此端口处监听；

当某个 Agent 要与自己所需的某项服务的提供者建立连接，则使用 QueryAddrByService(CString ServiceName，CString &IP，UINT &nPort)原语向 DS 查询提供 ServiceName 服务的 Agent 的 IP 地址和监听端口号。如果当前系统中不存在能够提供此项服务的 Agent，则 DS 会返回空的 IP 地址和监听端口号给发起查询的 Agent，如果当前系统中有 Agent 提供此项服务，则 DS 会从提供此项服务的 Agent 列表中选择首个 Agent 作为与发起查询请求的 Agent 建立连接的服务方，并将该提供服务的 Agent 的 IP 地址和监听端口号反馈给发起查询请求的 Agent。

②通信原语

BeginToListen(UINT nPort, ACCEPT_CALLBACK callback);Agent 调用该原语在端口 nPort 监听，同时注册自己的 OnAccept 回调函数，用于接收其它 Agent 的连接；

BeginToRequest(UINT &nCID, CString IP,UINT nPort);当 Agent 要与提供某项服务的 Agent 建立连接时，调用 QueryAddrByService 原语从 DS 获得了提供服务的 Agent 的 IP 地址和监听端口后，调用该原语向地址为 IP:nPort 的 Agent 发起连接请求，如果连接成功，返回该连接的 ID 标识 nCID；

PrepareForRecv(UINT nCID, RECEIVE_CALLBACK callback);当监听方接收新的连接时或请求方建立连接成功时，调用该原语建立连接标识为 nCID 的连接的接收回调函数，可以在该回调函数中对接收到的数据进行处理；

PrepareForClose(UINT nCID, CLOSE_CALLBACK callback);当监听方接收新的连接时或请求方建立连接成功时，调用该原语建立连接标识为 nCID 的 OnClose 回调函数，用于响应通信的对方关闭连接或结束对话的事件；

EndOfSession(UINT nCID);当连接的某一方主动要求断开连接，停止对话时，调用该原语，主动要求断开标识为 nCID 的连接；

Put(UINT nCID, CString strMsg);当 Agent 要发送消息给对方时，使用 Put 原语，将消息发送给对端，发送该消息的连接标识为 nCID。

③回调函数

typedef void(CAgent::* ACCEPT_CALLBACK)(UINT &CID);Agent 在监听时注册该回调函数，当该监听 Agent 收到连接请求时并成功接受连接后，该回调函数会返回所建立连接的 ID 标识 CID；

Typedef void(CAgent::*RECEIVE_CALLBACK)(CString &strMsg, UINT &CID);

Agent 在接收其他的 Agent 连接后或自己请求与其他某个 Agent 连接成功建立后，注册该接收回调函数，当有数据

到来时，该回调函数会返回接收到的数据 strMsg 和接收数据的连接标识 CID，通过 CID 能够辨别数据的发送方；

typedef void(CAgent::* CLOSE_CALLBACK)(UINT &CID);Agent 在接受其他的 Agent 连接后或自己请求与其他某个 Agent 连接成功后，注册响应通信对端主动断开连接的回调函数，当通信中某条连接的对端端开连接时，该回调函数会返回断开连接的标识 CID。

3 应用模块设计

该系统的功能主要是一个英文学习的人机对话系统，比如用户开始一个话题，这个话题用户可以从界面录入也可以直接通过麦克风说话，机器按照这个话题往下谈论，基本能够正常对话，该系统与以往的学习系统不同的是它可以在单机上运行，这种情况下适合个人学习，但它强大的功能主要体现在网络的运行上，例如用户可以和系统中的另外的机器对话，这样比较适合群体学习或者远程教学，所以这里主要说明该系统运行在网络的情况。下面简单介绍一下各模块 Agent 的划分与实现。

Agent 的基本要求就是功能相对独立，因此把相对独立的功能都做成一个 Agent，同时一个 Agent 要完成的任务也不是很多，一般一个 Agent 也就一个主要功能，这样可以增加自治性、降低耦合度。但又不能划分得过于详细，否则将导致 Agent 与 Agent 之间的耦合度过高，在增加系统通信量的同时，也增加了系统的不稳定性和复杂性，那也就违背了使用 Agent 技术开发系统的初衷。根据这个 Agent 划分原则，本系统中设计了 4 个功能模块：语句输入 Agent，语音识别 Agent，数据匹配与理解 Agent 和机器发音 Agent。

3.1 各功能 Agent 设计

(1)语句输入 Agent：一个输入界面接收用户的手工输入，然后把该用户输入的信息发送给数据匹配与理解 Agent 进行后续处理，处理后的结果还可以在语句输入 Agent 中显示出来。

(2)语音识别 Agent：系统接收用户直接语音录入，该 Agent 用于实时捕捉说话者的语音。与语句输入 Agent 一样，该 Agent 在获得了语音基本信息后，要跟系统中的知识库进行比对，看是否是自己能够识别的词汇，是则进行信息转换，否则放弃该信息；然后传递给数据匹配与理解 Agent 进行后续处理。在启动和退出语音识别 Agent 后，它也会自动启动和关闭语音识别引擎。

(3)数据匹配与理解 Agent：这是系统的一个核心，用于对获得的各种信息进行处理。对语句输入 Agent 和语音识别 Agent 送过来的词句，该 Agent 分析该句子的语法并进行语句匹配，将处理的结果交给语句输入 Agent 和机器发音 Agent。

(4)机器发音 Agent：它接收从数据匹配与理解 Agent 发来的信息，根据信息内容从词库中选出该语句需要的单词继而组成句子，然后启动语音合成引擎把该组合后的句子读出来。

3.2 系统运行流程

(1)启动 DS，Agent 进行注册，一般注册都发生在系统刚启动时，当有某个 Agent 由于某些原因停止服务或运行后，经过一段时间又重新启动，这时该 Agent 也必须向 DS 进行信息注册。因此以下各个 Agent 在启动时，做的第一项工作就是向 DS 注册，表明自己的存在和自己当前的位置。DS 所在地址可以作为初始化信息，当其它 Agent 启动时自动获取该地址。

(2)启动 Container，使之处于运行状态，所有参与计算环境的主机上都需要运行一个叫做 Container 的守护进程(或者服务)。Container 为运行在同一主机的 Agent 提供了一个统一的管理点，Agent 只需要和 Container 进行通信即可。

(3)语句输入 Agent 和语音识别 Agent，在启动语句输入 Agent 后，系统进行初始化等待用户的输入；在启动语音识别 Agent 的时候，系统会自动启动语音识别引擎，使引擎处于工作状态，监视硬件相应的端口。

(4)启动数据匹配与理解 Agent 准备进行系统工作。

(5)启动机器发音 Agent，系统启动语音合成引擎，使引擎处于工作状态，等待输出。

各个 Agent(除 DS)在向 DS 注册信息的同时，获取其它 Agent 注册信息，同时向其它已注册 Agent 发送消息进行信息联络，其它 Agent 回复消息，同时在自己的信息库中查找是否已经注册过该 Agent 的信息，如果没有注册则进行信息注册。当整个系统都运行完毕的时候，系统就可以开始正常工作了。

4 系统实现和测试

系统在 VC++环境下调试实现，对运行结果进行分析测试，首先进行功能测试，系统基本实现了人机对话的功能，最后对各个 Agent 间的通信和合作情况进行测试，各个 Agent 功能都是独立的，这里选择语句输入 Agent(订阅和接收消息)和数据匹配与理解 Agent(发布消息)，依据测试计划，对系统进行了包括注册、订阅消息、发布消息、退出等所有功能的压力测试，使用 3 台计算机，每台主机上启动一个数据匹配与理解 Agent 和两个语句输入 Agent，并让各台主机上的数据匹配与理解 Agent 同时向语句输入 Agent 发布消息(语句输入 Agent 向数据匹配与理解 Agent 订阅的消息)。每条消息采用定长 100B，一次连续发送 10 000 个数据包，共发送 200 000 个数据包；每条消息采用定长 5 000B，一次连续发送 1 000 个数据包，共发送 20 000 个数据包，语句输入 Agent 的界面上有一个计数器记录总共收到的数据包个数。测试结果表明，所有的语句输入 Agent 均能够正确接收所有数据匹配与理解 Agent 发布的消息，并且没有出现任何阻塞，所有进程能够正常通信。

5 结语

以改进人机对话任务单一和信息共享为目标，本文研究的基于 Agent 的人机对话系统具有如下新特性：在 Agent 封装下，人机对话系统可以实现多任务，各 Agent 能主动发现服务，进行自发互操作，多 Agent 之间能够相互协调和协同工作，克服以往人机对话系统任务单一化的缺陷，另外本系统容易扩展，如果要增加或改变系统功能，只需添加或改变相应的功能 Agent 即可，提高了系统的通用性。试验表明，本文设计的模型是有效可行的。

参考文献

1 董红斌, 王建华. 多 Agent 技术研究[J]. 计算机应用研究, 1999, 16(10): 29-30.

2 谢伟凯. 智能空间关键支撑技术的研究[D]. 北京: 清华大学, 2003.

3 李巧云, 李健东, 康 卓, 等. 开放分布式对象系统中的通信智能体[J]. 软件学报, 1997, 8(3): 235-240.

4 刘海燕, 王献昌, 王兵山. 多 Agent 系统的研究[J]. 计算机科学, 1995, 22(2): 57-61.