计算机科学 2002Vol. 29№. 10

基于多 Agent 的 Teamwork 研究综述*)

The Survey on Teamwork Technique Based on Multi-agent

王 玥 陈世福

(南京大学软件新技术国家重点实验室 南京 210093)

Abstract With the increasing popularity of the use of multi-agent systems in a variety of complex, dynamic domains, they become imperative to build an efficient means to help multi-agent cooperate and communicate well. Teamwork is one solution to this problem. This paper presents a survey on Teamwork technique, that includes illustrative domains, Teamwork theories and Teamwork models. Finally the paper points out the future development direction of Teamwork.

Keywords Teamwork, Multi-Agent, STEAM, Role

1 引言

基于多 Agent 的 Teamwork 技术是当今人工智能、计算机软件等领域研究的热点之一,具有广阔的应用前景。有关多 Agent 技术的研究在国内外已经得到迅猛的发展。研究多 Agent 技术其目的就是要让多个 Agent 能相互合作,解决现实中的问题。在实际生活中,我们面临的常常是一个动态、实时的环境,也就是说这个环境是不确定的。如果把多个 Agent 看成一个团队,则这种不确定性决定了不能预先把任务分配给团队中的每个 Agent。可以想见,只有建立一种机制提高 Agent 之间的团队合作能力,使团队成员间能根据环境的变化灵活地合作、通信才有可能解决这一问题。Teamwork 技术正是适应这一需要而被提出的。

Teamwork 就是指通过团队成员间的合作,分配任务以降低任务的复杂性,更好地利用资源,增强健壮性,来完成一个共同的目标。

目前,基于多 Agent 的 Teamwork 的构建技术在美国、日本等发达国家已经进行了多年研究并取得了一些成果。在不同的应用领域,国外的研究者们对 Teamwork 的构建进行了有效的尝试。例如 Milind Tambe[1,3]提出了一种通用的、已实现的 Teamwork 的模板 STEAM,Agent 利用这种模板来对通讯和合作进行自治的推理,这样可以实现团队的灵活性,并且,这种模板的创建可以实现跨应用领域的团队重用;又例如 Simon Ch'ng 和 Lin Padgham[5]在机器人足球领域设计了一种仅需要少量通讯或者零通讯的 Agents 柔性合作机制。他们建立了一支由具有角色特征(Role)、相应责任和策略的 Agents 构成的团队(Team)。而 Marcus J. Huber 和 Tedd Harley[4]通过一个复杂的、实时的、多玩家、多团队的互连网游戏:Netrek 对团队合作进行实验,他们使用了基于 PRS (Plan Recognition Scheme)的 Agent 的实现方法来提高团队内部、团队之间的协调能力。目前,基于多 Agent 的 Teamwork 技术已经成为 AAAI,IJCAI 等国际会议的重要内容。基于 Team 技术的机器人足球技术也成为研究的热点,而在一些国际刊物的 Agent 专刊中 Teamwork 的技术研究也成为重要的组成部分。国内对 Agent 系统规划问题、Agent 之间的通讯机制等进行了探讨,对多 Agent 机制也进行了研究。尽管目前面向多 Agent 技术的研究呈现出一派繁荣景象,但对如何在一个复杂的、动态的环境中构建一个有效的基于多 Agent 的 Teamwork 的研究还很不成熟,因为复杂动态环境的不确定性阻碍了一个有效团队的构建。例如:团队中的成员对环境的认识通常是不完全的,甚至有可能是不正确的。因此团队中的成员通常要对环境的变化进行快速的反应。仅仅通过对团队进行预先行动布置通常导致任务失败。这就要求我们建立起一种基于少量通讯的团队合作机制,这种团队机制能对环境的复杂变化做出正确的快速的反应。在现实生活中,很多任务不能仅仅通过单个 Agent 或是多个无组织的 Agent 来完成,而需要一支基于多 Agent 的团队才能完成,这就迫切需要对基于多 Agent 的 Teamwork 的技术进行研究。因此,该领域的研究具有重要的理论意义和应用价值。

本文主要对 Teamwork 技术目前研究的主要内容、现状和发展趋势进行了综述,并给出了一些系统实例。

2 Teamwork 场景及相关问题

要建立实际的 Teamwork 场景就必须先区分什么样的情况下可以称作是 Teamwork,什么情况不是。这里我们举一个简单的例子。一个玩具工厂里有很多工人,他们组成了一个团队,但是如果每个工人负责制作一个玩具的所有工序,也就是说工人之间并没有任何交流或需要协作的地方,则虽然他们也是一组人,我们也不能称他们这样的小组为 Teamwork。只有团队中的成员要交流,体现了合作性,才能称之为 Teamwork。上面的例子如果改成工人是流水线作业,可以算是最简单的 Teamwork。

当然我们想要研究的问题比上面的例子复杂得多。在实时、动态的环境中,Agent 间需要通信来实现合作。但是环境的状态变化是如此之快,如果有通信必然会影响执行效率,所以又希望尽量减少通信,究竟如何处理好这些问题实现团队高效合作就是 Teamwork 的任务了。下文将介绍现今几个研究较多的 Teamwork 场景:RoboCup,仿真直升机作战,以及一个游戏环境来说明在 Teamwork 设计中将面临的主要问题。

*)本文得到国家自然科学基金资助,项目号:60103012。王 玥 硕士研究生,主要研究方向为分布式人工智能、多 Agent。陈世福 教授,博导,主要研究方向为人工智能、机器学习、图像处理。

2.1 RoboCup

RoboCup 仿真足球[1,3,5]比赛是在一个标准的计算机环境内进行的，比赛规则与国际足联的比赛规则基本一致。这项比赛希望达到的目标是到 2050 年机器人球队将能战胜人类冠军球队。比赛采用的是 Client/Sever 方式。由 RoboCup 联合会提供标准的 SoccerServer 系统作为虚拟场地，它能允许竞赛者使用各种程序语言编写各自的 Client 程序，对比赛进行仿真，以离散的方式控制比赛过程。每个 Client 只能控制一个 Agent(球员)，相当于球员大脑。

这个环境具有真实世界问题的很多特点：环境高度动态，需要 Agent 间高度合作，且 Agent 间实现合作有多种方法，同时问题是有限制的(有足球规则)，目标和成功也定义明确。

在 RoboCup 中，Teamwork 要解决的问题就是实现多 Agent 间的合作，让所有的 Agent 具有一个共同的目标就是赢球。同时每个球员又有各自的目标，比如前锋要进球，后卫要防守等，Teamwork 要能协调好局部目标与全局目标，个体目标与共同目标间的冲突问题。

2.2 仿真直升机作战

这个环境[1,2]模拟的是战场，可用作军事训练(南加州大学，Tambe 等，1995)。这个环境中 Agent(飞行员)参与大规模的实战演习，甚至参与有人类专业飞行员的演习。研究者们给这些飞行员 Agent 提供了一个三维场景，有山，水，涵洞等各种地形。图 1 是 Tambe 等给出的战斗模拟图[1,2]。

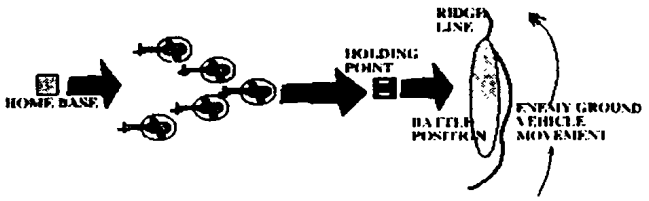

图 1 仿真直升机模拟图

这个环境也具有实时、动态环境的很多特点，比如 1)复杂性，飞行员不清楚敌机的位置、型号等。2)动态性，敌对双方直升机都是移动作战。3)不确定性，如同伴飞机有可能坠毁。4)可中断性，有些行为会因为情况的改变而终止。5)目标的可变性等。

从以上特点可以看出，仿真直升机环境是个相当复杂甚至比 RoboCup 还复杂的环境。要使 Agent 飞行员成功完成任务，至少要做到 3 点。第一，要让每个 Agent 飞行员知道还有哪些 Agent 参与了任务，即给每个 Agent 提供一个参与行动 Agent 列表。第二，要有好的合作规划，比如敌方飞机已经被击落，则击落敌方飞机的 Agent 要通知己方飞行员任务已经完成，可以返回基地。第三，Agent 应该有一定的阵形。比如，每两个 Agent 组成一个小组，双方互为搭档。这样即使两个 Agent 中有一个出事，另一个 Agent 还能代替它行动并通知己方别的飞行员 Agent 这一情况，并适当调整阵形继续保持这种搭档结构。

2.3 Netrek Agents 游戏系统

Netrek (Marcus J. H. &Tedd H.)[4]是一个复杂的、实时的、有多个游戏者、多个团队参与的互联网游戏，它可以通过互联网供全国乃至全世界的人游戏。这个游戏是采用 Server/Client 模式，环境由 Server 控制，世界各地的游戏者通过 Client 相连。最多可以有 16 个 Agent(游戏者)在一个 100000 * 100000 单位的网格上玩游戏。每八个 Agent 为一组。

游戏开始时把网格分为 40 块，成为 40 个行星(planet)，每十个又组成一个星系(galaxy)，有 4 个团队各占据一个星系。真正比赛时只有 2 支队伍参加，也就是说一次比赛只有 20 个行星或是说 2 个星系介入。每个行星有一种动态可变的资源——弹药，弹药可为行星自己生产，生产弹药的能力由行星的其他不变资源决定。游戏规则就是双方团队的 Agent 先战斗，战斗胜利者可把己方的弹药放在敌方的行星里，一旦一方的弹药超过原来行星上那方的弹药，则这个行星的归属权就变了，就变成弹药多的那方的领土了。团队的目标就是尽可能多地把敌方的行星占为己有。

在 Netrek 中，Agent 要做出许多决定，比如在什么地点，什么时间开始和对方团队中的 Agent 战斗，这些决定要快速、高效，受很多因素影响：某地点双方 Agent 的多寡(concentrations)，哪个行星正在有战斗发生，敌方的 Agent 是否携带有弹药，己方是否要防守自己的行星不被敌方的 Agent 放弹药以及己方团队的其他队员正在做什么等等。总之，就是要求通过 Teamwork 技术实现己方团队成员的合作，攻击敌方团队成员，对环境变化做出实时反应，并能确定共同的持续性目标。

3 Teamwork 理论

建立一个 Teamwork 模型要有相应的理论基础。这些理论基础不需要直接针对某一场景，而是要给 Teamwork 提供一个广泛的推理过程或启发式方法。在 Teamwork 发展的这些年中，已经有很多研究者提出了一些很有影响的理论，如 Joint Intentions 理论[6,1]，SharedPlan 理论[9,1]以及 Joint responsibility 理论[7,8]等。在这些理论的基础上，有的已经建立了实际的 Teamwork 模型，如依据前两个理论建立的 STEAM 模型[1,3](南加州大学)。下面我们对这 3 种理论加以简要介绍。

3.1 Joint Intentions 理论

所谓 Joint Intentions 理论[6,1]，就是指共同意图——团队的所有成员完成一个共同的任务，这个任务要被团队中所有成员认可，并一起通过协商争取实现。这个共同的任务又可称为 JPG(joint persistent goal)。如果团队用字母 Θ 表示，团队 Θ 的 JPG 要实现目标 p 可表示为 JPG(Θ, p,q)，其中 q 是无关变量。

形式 JPG(Θ, p,q)的使用不是任意的，必须满足 3 个条件：

①所有的团队成员都认为在当前状态下 p 尚未满足。

②所有的团队成员把 p 作为它们的共同目标，也就是说所有的成员都希望最终 p 能够实现。

③所有的团队成员共同认为：除非 p 已经被公认处于终止状态(实现、失败、不相关)，否则每个成员都将把 p 作为它们的 WAG(weak achievement goal)。用形式 WAG(μ,p,Θ,q)表示。这里的 μ 代表团队 Θ 中的一个成员。形式 WAG(μ,p,Θ,q)的含义是成员 μ 或者认为当前 p 为假，把 p 作为目标希望 p 最终为真，或者认为 p 已经处于终止状态，并要设法把它的这一信念成为整个团队的信念。

简而言之，Joint Intentions 理论就是让团队中的各成员把目标 p 作为 WAG，努力去实现，如果有一个成员认为 p 已经处于终止状态(实现、失败、不相关)，它就要设法让所有的成员都认可这一信念。其实 JPG 和 WAG 可以说是一样的，在大家没有公认实现目标的行为可以终止前，就是 WAG(局

部概念),形成对 p 的共识后,就是 JPG(全局概念)。

可以说,让所有团队成员都认为目标 p 已经可以终止是 Joint Intentions 理论的关键点。要实现这一点,要求团队成员能随时更新团队状态,建立较为完善的通信机制。

3.2 Shared Plan 理论

SP 理论[9,1]定义了一个新的基于意图的观点,称为 intending that——Agent 想要完成一个动作的意图。它与 Joint Intentions 理论的不同之处在于 Joint Intentions 理论是定义了一个共同的目标 JPG,而 SP 理论则通过一组公理或者说规则指导每个 Agent 的任务,包括确立 Agent 之间的通信,以使团队能共同完成某项行动。SP 理论中没有"我们要完成什么共同目标"的概念。

SP,其本质就是代表一个 Agent 团队(用 GR 表示)有一个共享的规划使团队中的 Agent 可以在一起完成某项行动 α。它可以根据是完全 SP(FSP)还是部分 SP(PSP)用递归的方法定义。从 FSP 和 PSP 的名字可以看出 FSP 与所有团队成员的规划相关。而 PSP 只与部分团队成员的规划相关。这里必须指出无论是 FSP 还是 PSP,都是指整个团队或是部分团队所共有的 SP(共享规划),与其所相关的个体成员规划是有很大区别的。SP 被用分布的方式映射到个体 Agent 的个别规划(individual plans)上,它包括完整的要完成行动 α 需要的所有工作。而个体 Agent 不知道也不需要知道整个团队的 SP,只须清楚它们自己的个别规划。

定义 SP 主要在于用分布式的方法分配团队中成员任务,并能让个体 Agent 知道其他 Agent 的状态以及共同合作完成行动 α 的能力。下面我们以 FSP 为例定义一个 SP。首先确定一个行动 α,它的各方面细节,即要完成的各项任务的总和(定义为 R_α)已经明确。R_α 中的具体任务用 β 表示。则形式 $FSP(P,GR,\alpha,T_P,T_\alpha,R_\alpha)$ 表示 Agent 团队 GR 在时间 T_P 的规划是根据任务集 R_α 在时间 T_α 完成任务 α。使用这个形式要满足的条件为:

①团队 GR 的所有成员共同相信 GR 要在时间段 T_α 处完成行动 α。

②GR 的所有成员相信 R_α 是实现行动 α 的任务集。

③对于任务集 R_α 中的具体任务采用递归定义的方法,即 GR 的子集 GR_t 用与定义 GR 相同的方法定义。

SP 理论希望能表达整个团队的意图和信念,而实际上,在动态、实时的环境中是很难形成真正意义上的 FSP,用得更多的是 PSP,制定的规划(plan)也往往是阶段性或局部性的,要最终形成 FSP,还要有相应的通信协议。

3.3 Joint Responsibility 理论

Joint Responsibility 理论[7,8]是 Jennings 等人提出的,他们认为只有目标(goal)或是只有实现行动的任务集都不足以保证合作问题的解决。他们采用形式和时序逻辑的方法建立模型,把责任(responsibility)看成 2 个层面。低层语言包括定义诸如目标、任务集、行动及相互依赖关系等。并用低层语言定义高层概念如针对共同目标和任务集的行为。

其实,Joint Responsibility 理论可以看成是前文提到的两种理论的某种程度上的一个结合。对于共同目标的处理,它参照了 Joint Intentions 理论的方法来表示共同目标,它的行为和 Joint Intentions 理论一样要基于 JPG。至于共同任务集,在 SP 理论中是用 joint activities 表示的,这里用的是 joint commitment,我们认为两者之间并没有很大区别。Jennings 提出个人任务行动(individual recipe commitment)的概念来定义每个团队成员需要完成的任务。这与 SP 方法中个体 Agent 的个别规划(individual plans)也是类似的。

总之,Joint Responsibility 理论要求所有的团队成员 $\alpha_1,\cdots,\alpha_n$ 能有一个共同的 JPG 去实现它们共同的目标,依据共同的任务集(用 $\sum$ 表示)执行各自负责的任务,每个 Agent 知道它们各自要做什么。Jennings 用形式 JOINT-RESPONSIBILITY($\{a1\cdots an\},\sigma,\sum$)定义 Joint Responsibility,其中 σ 与 Joint Intentions 理论中的 p 意义相同。

4 Teamwork 模型

随着 Teamwork 技术在多-Agent 领域的应用越来越广泛,有很多研究机构在这方面已经取得很好的进展,并根据 Teamwork 技术建立了实际的模型,运用在诸如 RoboCup 等仿真环境中,下面我们就以第 2 节中给出的场景,结合第 3 节中的理论,从概念、通信、Agent 结构 3 个方面各有侧重地介绍一些已有的 Teamwork 模型。

4.1 RoboCup 模型

近几年 RoboCup 大赛均和国际人工智能联合大会 IJCAI 会议同时召开,使得它的影响越来越大。所以很多基于 Teamwork 技术的模型被建立起来。其中最有影响的是两类[3],一类是所谓的场景独立 Teamwork 模型(domain-independent teamwork model),以 USC 的 Tambe 等人的 STEAM 模型[1,3]为代表。另一类是场景依赖 Teamwork 模型(domain-dependent teamwork model),以 RMIT 的 Ch'ng 和 Padgham 的模型[5]为代表,下面分别对 STEAM 和 Ch'ng 与 Padgham 模型给予简单的叙述。

4.1.1 STEAM 模型 STEAM 模型[1,3]的场景独立性是指与团队相关的责任是整个与场景独立的 STEAM 系统的一个部分。它是建立在 Joint Intentions 理论和 SP 理论基础上的。需要 Joint Intentions 理论的原因是共同意图可以提供给团队一个合作和交流的框架,这个框架能解决团队崩溃的情况,能给团队活动提供检测和支持的保证,还能给团队的活动给予明确的表示。用 SP 理论的原因是因为光有实现团队共同目标 p 的共同意图并不能解决例如团队成员行动一致的问题。

STEAM 模型基于一个分层次的交互式规划。它包含一个新概念:团队操作(team operators),其实就是一种交互式团队规划。当 STEAM 中的 Agents 选择一个团队操作执行时,他们将团队的共同意图具体化(体现场景的独立性)。再由团队操作明确表达团队的共同任务,而不仅仅是某个 Agent 个别任务。团队操作和个体操作一样也要有:先决条件规则,应用规则和终止规则。一个操作究竟是团队操作还是个体操作是动态决定的。当某个 Agent 要触发一个操作(operator)执行时,这个操作被注释为正在执行的 Agent,STEAM 模型动态地决定这个操作是作用在一个个体 Agent 上,一个子团队(团队中成员的子集)上,还是作用在整个团队上。如果是后两者,则这个操作就是团队操作。若是此操作只作用在触发它的 Agent 自己身上,则这个操作是个体操作。也就是说团队操作并不是预先定义的,而是在执行时动态决定的。

一个基于 STEAM 的 Agent 运用个体操作来支持它自己的当前状态,运用团队操作去支持整个团队的状态。团队的状态包括诸如当前的团队成员信息、成员间可能的通信方式、预先决定的团队领导等等。Agent 间是通过通信来建立和终

止团队操作的。

根据 USC 的实验，在 RoboCup 中使用 Teamwork 技术也就是使用 STEAM 模型能大大提高团队效率。

4.1.2 Ch'ng & Padgham 模型 这一模型[5]的场景依赖性是指它对团队中的每个 Agent 既指定了团队层次的责任，又指定了个别的责任。这个模型仔细分析 Agent 所能扮演的角色，Agent 根据预先定义好的策略动态的选择扮演或放弃角色。它牵涉到一系列概念：

①角色和责任：一个角色定义了一个个体 Agent 在游戏情节中选择扮演的部分，比如说，守门员就是足球游戏中的一个角色。一旦一个 Agent 承担了一个角色，它就有了一系列的责任，这些责任作为角色的一部分。

②策略和策略组集：确定角色后还要选择一个特别的策略以完成依赖于 Agent 角色的责任。一个策略可能牵涉一个或多个角色，多个角色构成策略组集。只和一个角色有关的策略称作独立策略，牵涉到多个角色的称作团队策略。团队策略通过和策略组集中的扮演角色的 Agent 相匹配来控制几个 Agent 之间的交互。

一个角色有名字、条件和一组责任。只有在条件满足的情况下一个 Agent 才能充当那个角色。比如一个 Agent 充当后卫，他的责任之一是防守对方球员的进攻。同时，别的 Agent 通过识别这个 Agent 的角色也选择一个角色使系统的性能达到最优，例如一个 Agent 充当了前锋的角色就必须有一个 Agent 充当中场球员的角色给前锋传球。

这个模型还有一个责任监控机制，监控环境中那些能激发角色的条件，并区分他们的优先次序。例如一个后卫可能会忽视一个盯人的责任而去实施一个更紧急的责任比如看到一个对方球员要进球了。当角色终止或是没有合适的角色时，Agent 可能保持不充当角色的状态直到一个新的角色被选择。

团队策略中的策略组集为 Teamwork 的表示提供了最底层系统的抽象。每一个 Agent 可以选择一个角色并且和别的角色一起执行一个团队策略。Agent 还要能改变他们的角色以适应环境或别的 Agent 扮演角色的需要。总之，在这个模型中 Agent 的角色选择机制非常重要。

4.2 仿真直升机作战模型

这一模型[1,2]也是建立在 Joint Intentions 理论基础之上，它的核心就是 Joint Intentions 框架。这个框架有两个主要方面。其一是明确定义了团队共同任务的性质，使所有团队成员相信这一任务目标尚未完成，并把这一任务作为大家共同的目标。其二是要保证团队中的所有成员不能随意从共同的任务中脱离出来。既然参与了这个团队，这个团队的共同意图就要求其成员必须承担相应的责任，除非目标终止。

无论是仿真足球环境还是仿真作战环境，通信都是一个很大的问题。在这种实时、动态的环境里，通信的代价往往很高。比如在作战环境下，通信可能会打破空中电波的平静，使团队暴露在不必要的危险中。就算通信的代价不高，持续发送信息也会造成很大的冗余和负担。对这一问题可以用先计算通信的利弊的方法，如果代价高于好处则或者采用不通信的方式或是继续等待环境状态发生变化，利大于弊的时候再执行。还有一种方式是通过团队成员的视觉而不是通信来更新当前团队状态。

由于此模型和 STEAM 模型依据的是同一理论，均由南加州大学的 Tambe 等学者定义，因此两者基本相似，其余可以参照 STEAM 模型，这里就不赘述了。

4.3 Netrek 模型

如第 2 节所述，Netrek 环境[4]具有动态、复杂、不确定性等特点。Netrek 规定游戏中的 Agent 在 1 秒钟内要做 5 个动作，这些动作包括对抗、移动以及通信等。在 Netrek 中 Agent 不可能长时间地不发出动作，而只是推理下一步该做什么。因为环境状态变化是如此之快，往往还没等 Agent 推理出结果，环境已经变得面目全非了。实际上，Netrek 要求其中的 Agent 要尽可能一直保持活动状态，在很短的时间内尽可能针对长期目标做出反应[4]。

在 Agent 的结构方面，Netrek 的研究者们认为很多结构都可以满足 Netrek 的要求，所以他们也尝试了好几种结构。比如过程推理系统结构 PRS(Procedural Reasoning System)。这种结构先评估系统当前状态，由状态驱动规则，当条件满足时，Agent 就执行相应的动作。在 PRS 的基础上密西根大学的研究者们建立了 UMPRS 系统。它主要包括 5 个部件：一个数据库，表示当前世界状态；一个规则库；一组待实现的目标；一个意图结构；一个翻译器。该系统在应用时不断测试当前状态，动态地重新做出行动决定。

总结及展望 从上面的讨论中，我们看出，目前 Teamwork 理论和技术是人工智能学者研究的热点课题之一，已出现了很多可喜的成果，并已运用到 RoboCup 等领域，但是对于 Teamwork 构建机制尚未形成成熟的理论和技术，该研究领域仍处于实验探索的量变阶段。因此我们应该从多应用角度开展基于 Agent 的 Teamwork 的理论和技术的研究。而现在 Teamwork 研究的热点和主要解决的问题是：

1)理论问题：Teamwork 模型建立在什么理论基础之上。可以看出，现在的主流思想是既要有团队共同的目标，也要有团队共同的规划。团队中所有成员要有共同的信念：当共同目标没有实现时，就要一直承担共同规划中它该承担的任务。

2)概念框架：Teamwork 模型中要用到的概念。就文中给出的几个模型来看，都要有角色、责任等概念。Agent 要能够动态地选择角色，承担和角色相关的责任，执行相应的动作。

3)通信问题：由于环境多是实时的、动态的，因此变化很快。不能有过多的 Agent 间的交换信息，同时有些环境也不适合有通信。现有模型常用估计通信代价的方法来决定是否通信。

4)Teamwork 模型和 Agent 模型：模型是构建 Teamwork 机制的基础，一方面要开展对模型理论的研究，我们需要知道运用这个技术是否能带来系统效率的提高。通过实验，比如 STEAM 模型在南加州大学的 ISIS 系统中的运用，已经很好地体现出了使用 Teamwork 技术比不用要高效很多。另外，Teamwork 技术只是作为一个技术运用在多 Agent 环境中的。所以还要建立相应的 Agent 模型。

5)失败问题：某个 Agent 失效，其他团队成员应该能及时知道，更新对当前系统状态的认识。

总之，Teamwork 技术对多 Agent 环境越来越重要，从训练、教育到基于 Internet 的信息集成以至未来的多机器人太空任务等都要用到它。就目前的研究看，这些模型大都还是建立在一个特定的预先定义好的环境中，比起现实世界的情况还算是简单的。今后的方向必然要求 Teamwork 更加灵活，能应付更复杂的情况，体现在团队中 Agent 要有学习的能力，对付失败的能力，以及更好地处理通信问题等方面。

目前，我们正在积极开展对基于多 Agent 的 Teamwork

计算机科学 2002Vol. 29№. 10

基于移动 Agent 的新型分布式文件系统研究*)

The Research of Mobile Agent-based Distributed File System

卢 军 卢显良 韩 宏 许 腾
(电子科技大学计算机学院 成都 610054)

Abstract In general, conventional distributed file systems is designed for LAN environment. They always play worse performance in WAN than in LAN. In this paper we present a new distributed file system - The Mobile Agent-based File System (MADFS). The intent of MADFS is to minimize the latency overhead inherent to the distribution of a large file system in WAN. MADFS organizes hosts into a hierarchical structure, uses Mobile Agents as the underlying transfer, communication and synchronization mechanism and uses hierarchical-and-converge cache coherency protocol to minimize network usage. MADFS can achieve better performance and availability than conventional distributed file system in WAN. In this paper, the architecture of MADFS, performance analysis, cache coherency protocol and security are discussed. At last of this paper, the disadvantage and future work of MADFS are presented.

Keywords Mobile agent, Distributed file system, Performance

1. 引言

现代分布式文件系统的发展趋势[1]是 Transparency、Scalability、Unix Semantics、Reliability、Adaptation 和 Security。随着技术的发展，人们希望建立一种分布式文件系统来管理更广范围的文件资源，例如管理 WAN 环境中的文件资源，甚至管理 Web 环境中的文件资源。传统分布式文件系统大多是针对高带宽、低时延的 LAN 设计的，不能适应低带宽、高时延的 WAN 环境[2]，主要体现在：(1)文件存储协议、Cache 管理机制不适合 WAN。传统分布式文件系统的文件传输协议、Cache 管理机制是针对高带宽、低时延的 LAN 设计的，在低带宽、高时延的 WAN 下性能很差。(2)伸缩性差。传统的单一中央文件服务器模型完全不适应 WAN 环境下的分布式文件系统，因为 WAN 环境下的分布式文件系统可能有成千上万个客户，巨大的文件数量和存储规模将使中央服务器不堪重负。(3)可用性差。WAN 中的分布式文件系统将在更广范围内容纳大量的服务器和主机，在这种情况下一台服务器或主机崩溃或者网络连接中断的可能性更大，因此新的分布式文件系统必须可以容忍一台服务器或主机暂时不可用或者离线。

Sprite[3]是用 RISC 进行符号处理课题组开发的分布式文件系统，Sprite 和 NFS 一样是基于 Remote Procedure Call (RPC)来实现的。RPC 是针对 LAN 设计的，在 WAN 中性能很差[4]，导致 Sprite 很难在 WAN 中应用。Coda[5]是卡同基梅隆大学开发的分布式文件系统，Coda 通过服务器执行 Call-back 来实现 Cache 一致性管理。虽然 Coda 采用了 AVSG (Access Volume Storage Group)机制来提高 Call-back 的执行效率，但是由于 Coda 的体系结构是单层的 Client/Server 结构，所有的 Call-back 工作必须由少数文件服务器担任，因此随着系统规模的增加文件服务器将成为系统性能的瓶颈。

本文提出了一种基于移动 Agent 的新型分布式文件系统—MADFS。MADFS 使用分层体系结构，将整个文件系统划分为若干域。每个域由域服务器管理，所有域服务器由主服务器管理。这种体系结构避免了少数中央服务器成为系统扩展的瓶颈。MADFS 使用移动 Agent 来实现数据传输、通信和同步机制，比使用 RPC 更适应在 WAN 环境中工作。MADFS 采用了高效的 Agent 代码加载机制和"分层汇聚"Cache 管理机制来降低在 WAN 中的网络开销。上述特色使 MADFS 拥有很好的伸缩性、适应性和可用性，比传统分布式文件系统更适应在 WAN 环境下运行。

本文介绍 MADFS 的体系结构之后介绍其高效 Agent 代码加载机制以及其中的 Cache 管理机制，并对其安全问题进行了分析，最后提出了未来的工作。

2. MADFS 体系结构

传统的分布式文件系统，例如 NFS、Coda、Sprite 采用一层的 Client/Server 模型。这种模型在 LAN 中工作得很好，但却不适合在 WAN 中工作，因为单层的 Client/Server 模型要

*)本文受国家 95 重点攻关项目支持。卢 军 博士生。卢显良 教授，博士生导师。韩 宏 博士生。许 腾 讲师。

技术的研究，并将研究的初步成果用于 RoboCup 仿真系统中，已取得很好的效果。

参 考 文 献

1 Tambe M. Towards Flexible Teamwork. Journal of Artificial Intelligence Research, 1997, 7: 83～124
2 Tambe M. Executing Team Plans in Dynamic, Multi-agent Domains. In: Proc. of the AAAI Fall Symposium on Plan Execution: Problems and Issues, 1994
3 Marsella S, et al. On being a teammate: Experiences acquired in the design of robocup teams. In: Proc. of the 3rd Intl. conf. on autonomous agents, Agents'99
4 Marcus J H, et al. Dynamic Environment: Autonomous Netrek Agents. In: Proc. of the First Intl. Conf. on Autonomous Agents (AA), Marina del Rey, CA, Feb. 1997. 332～339
5 Ch'ng S, Padgham L. From Roles to Teamwork: a framework and architecture, (1997) in Applied artificial Intelligence, special issue on Robocup, 1997
6 Cohen P R, Levesque H J. Confirmation and joint action. In: Proc. of the Intl. Joint Conf. on Artificial Intelligence, 1991a
7 Jennings N. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. Artificial Intelligence, 1995, 75
8 Jennings N. Commitments and conventions: the foundation of coordination in multi-agent systems. The Knowledge Engineering Review, 1994, 8
9 Grosz B, Kraus S. Collaborative plans for complex group actions. Artificial Intelligence, 1996, 86: 269～358