P2P MMOG中一种结构化多代理节点模型

罗 佳[1] 常会友[2] 衣 扬[1]
(中山大学信息科学与技术学院 广州 510275)[1] (中山大学软件学院 广州 510275)[2]

摘 要 针对大型多人在线游戏(Massive Multi-player Online Game)中不断增大的游戏资源需求同有限的服务器负载能力之间的矛盾,提出一种负载均衡的结构化多代理节点(Structured Multi-Agent)模型。该模型对P2P MMOG的相关理论进行了定义,并在此基础上构造出节点加入算法、邻居发现算法和节点跨域算法。理论证明这3个算法保证了资源状态的一致性。同时SMA将所有资源的处理权均衡地分配给兴趣域内的所有节点,实现了节点间的负载均衡。对模型伸缩性、响应速度和节点负载等方面的理论分析表明,SMA模型具有较好的性能优势。

关键词 大型多人在线游戏,对等网络,兴趣域,Pastry协议
中图法分类号 TP393 **文献标识码** A

Structured Multi-Agent Model for P2P MMOG

LUO Jia[1] CHANG Hui-you[2] YI Yang[1]
(College of Information Science and Technology, Sun Yat-sen University, Guangzhou 510275, China)[1]
(College of Software, Sun Yat-sen University, Guangzhou 510275, China)[2]

Abstract For the contradiction between the increasing requirement of resources and limited load capacity of servers in Massive Multi-player Online Game (MMOG), a Structured Multi-Agent (SMA) model with load balancing was proposed. The theory of P2P MMOG was defined for SMA and three core algorithms which contain node joining algorithm, neighbor discovery algorithm and cross domain algorithm were constructed. These algorithms theoretically guarantee the consistency of states of all resources. In order to complement the load balancing, SMA assigns the processing licenses of resources to all nodes in an Area of Interest (AOI). Through analyzing the scalability, response speed and load consumption etc., SMA has fine performance.

Keywords MMOG, P2P, Area of interest, Pastry protocol

传统的大型多人在线游戏(Massive Multi-player Online Game)主要依靠服务器来维持游戏世界的逻辑处理,其弊端在于服务器有限的负载能力难以满足日益增加的资源需求。因此,基于P2P的MMOG成为了新的研究方向。P2P MMOG主要面临以下问题:1)资源的存储问题;2)资源状态的一致性问题;3)节点的组织和管理方式;4)系统的稳定性;5)安全性[3]。本文将围绕这些问题展开讨论。

1 相关工作

近年来,针对P2P MMOG的研究主要集中在全分布式非结构化网络拓扑(Decentralized Unstructured Topology, DUT)和全分布式结构化网络拓扑(Decentralized Structured Topology,也称DHT网络)两个方向。DUT方面,主要采取节点间直接通信的方式,如lockstep[5],Asynchronous Synchronization[5],New Event Ordering[6],Secure Event Agreement[7],Efficient and Secure Event Signature[8]等。由于这些方法具有DUT的先天缺陷,节点间的组织较难维持,对象状态的一致性也得不到保证,因此更多的研究转向了DHT网络。

目前基于DHT的MMOG研究主要采用区域协作管理的机制,最具代表性的有Thorsten Hampel[1]和Bjorn Knutsson[9]等分别提出的在每个AOI内分配一个协作者,管理AOI内的所有节点。然后利用构建于Pastry[13]上的应用层多播Scribe来进行资源状态的分发。Takuji Iimura[10]提出了一种区域联合机制,每个AOI区域有一个管理节点和数据持有节点,管理节点负责执行逻辑判断和缓存AOI内的数据,并将数据备份到数据持有节点上,AOI内的其他节点直接与管理节点通讯。Jin Zhou[4]对Takuji Iimura[10]的方法进行了改进,它利用LENS(Locality Embedded Naming Strategy)算法来选择同AOI内节点通讯时平均延迟最小的管理节点。本文将以上这些方法统称为结构化单协作者(Structured Single Coordinator, SSC)模型。SSC模型的缺点是当AOI内节

到稿日期:2009-04-08 返修日期:2009-06-16 本文受国家自然科学基金项目(60573159),广东省自然科学基金重点项目(05200302),粤港关键重点突破项目(2006Z1-D6021)资助。

罗 佳(1983-),男,博士,主要研究方向为P2P与网络游戏,E-mail:is01lj@163.com;常会友(1962-),男,教授,博士生导师,主要研究方向为嵌入式系统、数据库、工作流、操作系统、大型制造业管理信息系统和计算机集成制造系统;衣 扬(1967-),女,副教授,主要研究方向为智能控制理论和算法、基于知识系统的非确定信息的处理等。

点数量不断增加时，协作者很可能达到性能瓶颈，同时没有解决相邻 AOI 内节点的交互问题。Anthony[2] 提出的 MOPAR 方法加强了邻居发现算法，且允许 AOI 内的节点直接通信，减少了协作者的负载，但也没有很好地解决资源状态的一致性问题。以上所有方法还存在以下问题：1）没有考虑共享资源的维护；2）没有解决资源在 AOI 间交互[12] 的状态一致性问题。本文提出的 SMA（Structured Multi-Agent）模型有效地解决了上述问题，并保证了良好的性能优势。

2 P2P MMOG 理论

定义 1（资源） 游戏中所有资源的集合记作 R，任意资源 r 是一个三元组 $\langle id,k,s\rangle$，id 是资源的唯一标识，k 是资源的关键字，s 表示资源的状态或属性集合。所有具有关键字 k 的资源集合表示为

$$R_k=\{\langle id,k',s\rangle \mid \langle id,k',s\rangle\in R \text{ and } k'=k\} \quad (1)$$

R_k 也表示关键字为 k 的 AOI 内所有资源的集合。由资源的全局唯一可知 $R_{key1}\cap R_{key2}=\emptyset$。为方便说明，记节点当前所在的 AOI 为 SAOI，与 SAOI 相邻的所有 AOI 记为 EAOI，SAOI 与 EAOI 共同称作 WAOI。

通过 Pastry 协议对资源 R_k 的资源定位公式为

$$\forall j\in N,\exists i\in N,|i-SHA\text{-}1(k)|\leqslant|j-SHA\text{-}1(k)| \quad (2)$$

其中，N 代表所有节点的集合，SHA-1 算法用于计算全局唯一的键值。式(2)表示由 Pastry 协议找到节点 id 与 SHA-1(k)最接近的那个节点。节点 id 由 Pastry 协议进行分配。

定义 2（事件） 所有资源之间的交互都是一个事件 e。根据事件作用资源的数目可分为原子事件（单一资源）和复杂事件（多资源）。任意复杂事件都可以拆分为多个原子事件的序列。因此在本文中规定所有事件都是原子事件。

定义 3（状态） 两个状态 s_i,s_j 相等的充分必要条件为 $i==j$ 且 s_i,s_j 中所有的属性都完全相等。

定义 4（状态机） 任意节点 i 上都运行着一个状态机 M_i，用于处理 WAOI 内的事件，M_i 的状态 GS_i 为 WAOI 内所有资源状态的集合，即

$$GS_i=\{s\mid\langle id,k',s\rangle\in(R_{k1}\cup R_{k2},\cdots,\cup R_{kn})\}=\{s_1,s_2,\cdots,s_n\} \quad (3)$$

其中，$k1,k2,\cdots,kn$ 为 WAOI 中所有 AOI 的关键字。这里注意区分节点的状态 s_i（节点也是一个资源）与节点状态机的状态 GS_i，前者包含于后者。

假设 1（状态转换） 资源的状态 s 在经过原子事件 e 后能且只能转换到状态 s'。同理可以推出节点状态机 M_i 的状态 GS_i 经过原子事件 e 后能且只能转换到状态 GS_i'，即

$$GS_i'=e(GS_i)=e(\{s_1,s_2,\cdots,s_r,\cdots,s_n\})=\{s_1,s_2,\cdots,e(s_r),\cdots,s_n\}=\{s_1,s_2,\cdots,s_r',\cdots,s_n\} \quad (4)$$

定义 5（一致性） 任意两个节点 i,j 上状态机的状态分别为 GS_i 和 GS_j，若 $GS_i=GS_j$，则称 GS_i 和 GS_j 一致。若 $GS_i\cap GS_j=\emptyset$，则 GS_i 和 GS_j 无关。若 $GS_i\neq GS_j$，且 $GS_i\cap GS_j\neq\emptyset$，则 GS_i 和 GS_j 部分一致。若 $i=j$，有 $s_i\neq s_j$，$s_i\in GS_i$，$s_j\in GS_j$ 则 GS_i 和 GS_j 不一致。

在正确情况下，任意节点同 SAOI 内所有其他节点状态机的状态都一致，SAOI 内的节点同 EAOI 内的节点状态机的状态保持部分一致，与距离较远的 AOI 间节点状态机的状态无关，不应存在不一致的情况。

定理 1（一致变换） 任意两个节点 i,j 上状态机的状态分别为 GS_i 和 GS_j。1）若 GS_i 和 GS_j 一致，则发生原子事件后的状态 GS_i' 和 GS_j' 一致或无关。2）若 GS_i 和 GS_j 部分一致，则发生原子事件后的状态 GS_i' 和 GS_j' 部分一致、一致或无关。

证明：1）由 GS_i 和 GS_j 一致可知，$GS_i=\{s_1,s_2,\cdots,s_n\}=GS_j$。若事件为加入事件，则

$$GS_i'=e(GS_i)=e(\{s_1,s_2,\cdots,s_{r-1},s_{r+1}\cdots,s_n\})=\{s_1,s_2,\cdots,s_{r-1},s_r,s_{r+1},\cdots,s_n\}$$

$$GS_j'=e(GS_j)=e(\{s_1,s_2,\cdots,s_{r-1},s_{r+1}\cdots,s_n\})=\{s_1,s_2,\cdots,s_{r-1},s_r,s_{r+1},\cdots,s_n\}$$

有 GS_i' 和 GS_j' 一致。若事件为删除事件，则

$$GS_i'=e(GS_i)=e(\{s_1,s_2,\cdots,s_{r-1},s_r,s_{r+1},\cdots,s_n\})=\{s_1,s_2,\cdots,s_{r-1},s_{r+1}\cdots,s_n\}$$

$$GS_j'=e(GS_j)=e(\{s_1,s_2,\cdots,s_{r-1},s_r,s_{r+1},\cdots,s_n\})=\{s_1,s_2,\cdots,s_{r-1},s_{r+1}\cdots,s_n\}$$

有 GS_i' 和 GS_j' 一致或无关（GS_i' 和 GS_j' 都为空）。若事件为改变事件，则

$$GS_i'=e(GS_i)=e(\{s_1,s_2,\cdots,s_r,\cdots,s_n\})=\{s_1,s_2,\cdots,e(s_r),\cdots,s_n\}$$

$$GS_j'=e(GS_j)=e(\{s_1,s_2,\cdots,s_r,\cdots,s_n\})=\{s_1,s_2,\cdots,e(s_r),\cdots,s_n\}$$

有 GS_i' 和 GS_j' 一致。

2）由 GS_i 和 GS_j 部分一致可得 $SS=GS_i\cap GS_j$，$LS_i=GS_i-SS$，$LS_j=GS_j-SS$。若事件为加入事件，则

$$GS_i'=e(GS_i)=e(SS\cup LS_i)=SS\cup LS_i\cup\{s_r\}$$

$$GS_j'=e(GS_j)=e(SS\cup LS_j)=SS\cup LS_i\cup\{s_r\}$$

有 GS_i' 和 GS_j' 部分一致。若事件为删除事件，则

$$GS_i'=e(GS_i)=e(SS\cup LS_i)=(SS\cup LS_i)-\{s_r\}$$

$$GS_j'=e(GS_j)=e(SS\cup LS_j)=(SS\cup LS_i)-\{s_r\}$$

此时，若 s_r 属于 SS，且 $SS=\{s_r\}$，则 GS_i' 和 GS_j' 无关。否则部分一致。若 s_r 属于 LS_i 或 LS_j，则 GS_i' 和 GS_j' 部分一致。若事件为改变事件，则

$GS_i'=e(GS_i)=e(SS\cup LS_i)=e(SS)\cup LS_i$，或 $GS_i'=SS\cup e(LS_i)$

$GS_j'=e(GS_j)=e(SS\cup LS_j)=e(SS)\cup LS_j$，或 $GS_j'=SS\cup e(LS_j)$

则 GS_i' 和 GS_j' 部分一致。证毕。

定义 6 任意节点同其 SAOI 内所有节点状态机的状态都满足一致，且同 EAOI 内所有节点状态机的状态都满足部分一致，则称此系统满足一致性。

3 SMA 模型体系结构

SMA 模型是一个负载均衡的结构化多代理节点模型。该模型以 AOI 为最小单位组织和管理资源。每个 AOI 通过 Pastry 协议分配一个管理节点，用于 AOI 内代理节点的管理和邻居节点的发现。所有资源的交互事件通过处理权分配算法均衡地分配给 AOI 内的代理节点处理，因此所有节点的计算能力能够得到充分利用。下面详细描述 SMA 模型的架构和核心算法。

3.1 SMA 模型的架构

如图 1 所示，SMA 模型由 Pastry 应用层、协作层、游戏逻

辑层构成。Pastry 应用层主要用于资源定位，即通过输入资源的关键字返回该资源所在的管理节点信息，因此同一 AOI 内的所有代理节点都可以利用 Pastry 定位到管理节点，从而建立起 AOI 内代理节点间的联系。协作层主要负责节点的管理和事件处理权的分配，是 SMA 模型的核心，包括了节点加入算法、邻居发现算法和节点跨预算法。后面将对这 3 个算法进行详细描述。游戏逻辑层则负责处理 AOI 内发生的事件。

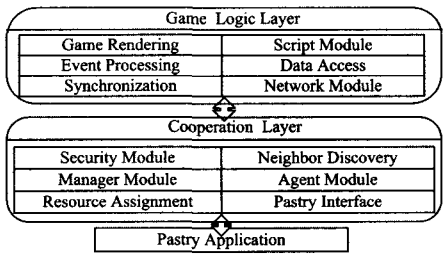

图 1　SMA 模型架构图

SMA 模型通过数据服务器[11]、管理节点和代理节点的相互协作来维持整个系统的稳定运行。3 类节点的职能描述如下。

1)数据服务器：负责用户的身份验证和信息保存；发布共享资源给节点。共享资源是指当节点还未加入时由服务器暂时维护的资源。

2)管理节点：负责邻居节点发现和维护代理节点链表。邻居节点发现能获取 EAOI 内的所有管理节点和代理节点信息，以便相邻 AOI 间的资源可以进行交互。代理节点链表保存着每个 EAOI 内的代理节点信息。任意节点通过这个链表都可以获取到 WAOI 内的所有代理节点。

3)代理节点：分配事件的处理权，将事件发送给具有处理权的代理节点处理，事件处理完成后将改变的资源状态更新给 AOI 内的所有其他代理节点。

可以看出，管理节点也要兼备代理节点的功能。图 2 表示两个关联的 AOI 之间节点的相互连接情况。圆点代表代理节点，六角星代表管理节点。

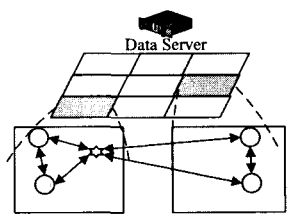

图 2　节点关系图

图 2 中，世界空间被划分为许多方形的 AOI，当然也可以用其他多边形来划分，甚至是用三维多面体来分隔。由于管理节点所在的 AOI 与其所管理的 AOI 没有任何关系，因此管理节点任意改变其所处的 AOI 时，其管理下的 AOI 内所有代理节点同它的关系保持不变。图 2 只表示了 2 个 AOI 之间节点的连接关系。实际上 SMA 模型可以形成一个庞大稳定的节点关系链，它依靠下面的 3 个核心算法来实现。

3.2　节点加入算法

节点加入算法是 SMA 模型的基础算法。通过它来建立起节点之间的联系，进而形成一个庞大的节点关系链。下面是节点加入算法的步骤。

1)获取管理节点：当任意节点 i 加入时，首先连接数据服务器进行身份验证和获取用户最后一次离线时的状态信息。然后将节点 i 所在 AOI 的关键字提交给 Pastry 协议层，由 Pastry 协议层定位该 AOI 的管理节点。随后将自己加入到管理节点的代理节点链表中，并设置自己为代理节点。

2)获取资源：若定位到的管理节点是新的管理节点，表明 AOI 内还未有节点加入，于是从数据服务器获取 AOI 内的共享资源。否则，从管理节点处获得所有代理节点的信息并直接从这些代理节点处获取共享资源。

3)资源处理权的分配：当节点成功加入后，需要重新分配资源的处理权，以便 AOI 内任意资源都有且只有一个代理节点处理该资源的交互事件。事件处理完成后，改变的状态被更新给 AOI 内的所有节点。

4)资源转移：管理节点必须定时检查是否有节点满足资源定位式(2)的条件。若有，则此节点将成为新的管理节点，并与之交换代理节点链表信息。下面是这一算法的伪码。

算法 1　*Node-Joining Algorithm*

```
Step1  InitClient( _sid );
Step2  _new_m = Pastry( _saoi_key );
Step3  AddToAgentList( _sid );
Step4  if (IsManager(_new_m ))
          SetManager( _new_m );
          GetResFromServer();
Step5  else
          GetResFromAgent();
Step6  ReallocateRes();
Step7  while(1)
          CheckNewManager();
          ProcessEvent();
          UpdateToAOI();
```

算法 1 中资源处理权的分配公式为

$$\forall j \in N_k, \exists i \in N_k, |i - SHA\text{-}1(id)| \leqslant |j - SHA\text{-}1(id)| \tag{5}$$

其中，N_k 是关键字为 k 的 AOI 内所有节点的集合。式(5)表示资源的处理权只能分配给资源所在 AOI 内与 SHA-1(id) 最接近的那个代理节点。当然，资源的处理权不一定要在节点加入完成时就全部分配好。相反只需在资源交互时，由代理节点来计算所要交互资源的处理权即可。下面是资源处理权的分配步骤。

1)节点信息更新同步：当节点 i 成功加入后，直接通知 AOI 的所有代理节点。这些代理节点收到消息后停止事件的处理，但允许接收消息并缓存到特殊处理区。所有代理节点向 AOI 内其他节点发送节点变动确认消息。当任意代理节点收到所有其他代理节点发送的确认消息后，便可继续进行事件的处理。

2)处理权分配机制：资源处理权的分配由所有的代理节点分担。即节点访问资源时，先根据式(5)计算该资源的处理权，然后缓存(避免重复计算)并发送给具有处理权的代理节点处理。存放在特殊处理区的消息将被优先计算其处理权并转发到正确的代理节点。两个节点之间的交互事件被直接交给交互的目标节点处理。

3)共享资源的分配：由于共享资源有自己的 AI 行为(如 NPC)，这些行为需要代理节点来维持，因此也要对这些资源分配处理权。幸运的是，不需等待这些资源的处理权分配完资源就能进行交互。因为处理权已经由分配算法决定了，交

互事件先于行为事件在游戏中对用户的体验不会造成影响，因此可以先处理交互事件再处理行为事件。

算法 1 虽然建立起了 AOI 内节点间的连接，但它只解决了 SAOI 内的节点发现和资源交互问题，没有解决资源间的跨域交互，因此需要邻居发现算法来获取 EAOI 内具有资源处理权的代理节点。

3.3 邻居发现算法

邻居发现算法建立了相邻 AOI 间节点的联系，实现了世界范围内的无缝交互以及节点跨域时的平滑过渡。下面是该算法的描述。

1)邻居发现的前提：代理节点 i 完成算法 1 的加入过程后，如果 i 是 SAOI 内唯一加入的节点，则需要进行邻居节点发现。否则直接从管理节点处获取 WAOI 内的其他代理节点信息并获取邻居共享资源。

2)节点发现过程：管理节点通过 Pastry 协议层获取到 EAOI 内的所有管理节点，从而获取到所有的代理节点信息。资源跨域交互时仍然根据式(5)的处理权分配方法将资源交给其 AOI 内相应的代理节点处理。

3)邻居共享资源的分配：若 EAOI 中部分 AOI 还没有节点加入，则这些 AOI 内就不会有共享资源，也就不存在跨域交互。事实上这是不符合游戏要求的。解决办法是，当 SAOI 的管理节点定位到该 EAOI 的管理节点时，将 SAOI 中的所有代理节点加入到此 EAOI 的管理节点的代理节点链表中，并从数据服务器将资源下载到 SAOI 的代理节点上。任意资源与该 EAOI 内的资源交互时，从该 EAOI 的管理节点处获取到代理节点信息，并通过式(5)的处理权分配方法将事件交给相应的代理节点处理。

4)邻居共享资源的恢复：当邻居 AOI 内有新节点加入时，由邻居 AOI 的管理节点替换所有的代理节点信息，并从 SAOI 内的代理节点处获取邻居 AOI 内的共享资源。获取过程仍然遵循资源处理权重分配方法。

5)状态更新：任何事件在处理完成后，都由代理节点将资源状态更新给 WAOI 内的所有其他代理节点。

算法 2 Neighbor Discovery Algorithm

```
Step1  _eaoi_m = Pastry( _eaoi_key );
Step2  if ( ! IsManager( _eaoi_m ))
           SetManager( _ eaoi_m );
           AddAgentToManager();
           ReleaseResFromServer();
           ReallocateRes();
Step3  else GetAgentFromManager();
           GetResFromAgent();
Step4  while( 1 )
           ProcessEvent();
           UpdateToWAOI();
```

这里对算法 2 中关于邻居共享资源的分配方法进行一些补充，如图 3 所示。

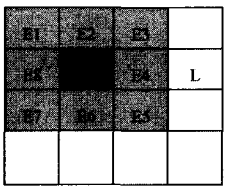

图 3 邻居关系图

S 表示 SAOI，E 表示 EAOI。假设这 16 个 AOI 内没有节点，则当任意节点加入 SAOI 时，首先执行节点加入算法，然后进行邻居发现。由于 EAOI 内没有任何节点，因此从数据服务器下载 EAOI 内的所有资源给 SAOI 内的这个节点，即整个 WAOI 的资源处理权都分配给这个新加入的节点。这时有 3 种情况发生：

1)SAOI 内加入新节点。此情况只需在资源交互前重新计算资源的处理权。

2)EAOI 内加入节点。以 E4 为例，此时节点先进行节点加入算法，并从管理节点处获得 E4 的代理节点信息，从而获得 E4 内所有的共享资源。然后从自己的管理节点处删除旧的代理节点信息，加入新的代理节点。最后进行邻居发现。

3)L 内加入节点。完成节点加入算法后，执行邻居发现算法，得到的 E3，E4，E5 的代理节点实质为 S 的代理节点。

以上 3 种情况表明资源在跨域交互时，无论 AOI 是否有节点加入，都能够找到处理该资源的代理节点。所以算法 2 的共享资源分配方法能够保证任意相邻 AOI 内资源的交互。

3.4 节点跨域和失败算法

节点跨域和失败时都会引起节点关系的变化，进而影响 SMA 模型的稳定性。因此跨域和失败算法应该尽量保证节点在较短的时间内恢复到事件处理状态。下面是该算法的描述。

1)节点跨域处理：因为管理节点也具备代理节点的功能，所以两者跨域时的处理相同。若新的 AOI 内已经有节点加入，则直接通知新旧 AOI 的代理节点执行算法 1 中资源处理权重新分配方法的步骤。否则遵循邻居共享资源分配方法的第 2 种情况加入节点。由于此节点本身已经有新 AOI 的共享资源，因此只需由管理节点替换代理节点信息并进行邻居发现。

2)节点失败：将失败节点的信息保存到数据服务器，并进行资源处理权的重新分配。若该节点是管理节点，且无论管理节点所管理的 AOI 是否有节点加入，都由此管理节点中的代理节点(可能是邻居 AOI 的代理节点)根据 Pastry 协议重新定位到新的管理节点，由新的管理节点进行邻居代理节点链表更新。

算法 3 Cross-Area and Failure Algorithm

```
Step1  if (CrossAoi( _sid ))
           ChangeAgentFromSAOIManager( _sid );
           if ( HasNode( _eaoi_id ))
               ChangeAgentFromEAOIManager( _sid );
               ReallocateRes ();
           else
               ReplaceAgentBySelf( _sid );
               NeighborDiscovery();
Step2  else if (Failure( _sid ))
           SaveInfo( _sid );
           ReallocateRes ();
           if ( IsManager( _sid ))
               _new_m = Pastry( _saoi_key );
               SetManager( _new_m );
               AddAgentToManager();
```

4 SMA 模型评价

4.1 一致性

定义2的原子事件、定理1的一致性变换以及SMA模型的工作机制为保证所有资源状态的一致性提供了理论上的依据。于是有如下定理成立。

定理2 SMA模型能够保证所有资源状态的一致性。

证明:根据定义6的一致性判定标准有:

1)当任意节点 i,j 位于SAOI时,由算法1可知,后加入的节点可以从已加入节点处获取所有资源的状态,因此节点 i,j 状态机的状态是一致的。当SAOI内发生事件 e 时,由 e 的原子性和定理1可知,节点 i,j 状态机的状态在经过 e 后应该保持一致。而SMA中的资源处理权分配方法和状态更新两个步骤达到了这一要求。资源处理权分配方法保证了资源处理权的唯一,从而保证了资源状态的改变是唯一的,更新状态使得资源状态改变的唯一结果作用到所有节点的状态机上。因此节点 i,j 状态机的状态能够保持一致。

2)当任意节点 i,j 位于相邻AOI时,由算法2可知,相邻AOI之间的代理节点会相互更新资源状态,因此节点 i,j 状态机的状态满足部分一致。当AOI内的事件 e 发生时,由 e 的原子性和定理1可知,节点 i,j 状态机的状态在经过 e 后应该满足部分一致。同样,SMA采取的资源处理权分配方法和状态更新两个步骤达到了这一要求。因此 i,j 状态机的状态能够保持部分一致。

综上所述,SMA模型能够保证所有资源状态的一致性。证毕。

定理2的证明过程存在一个前提条件,也就是当节点变动时,资源处理权分配方法能够保证资源处理权的唯一。于是有了如下定理。

定理3 资源处理权分配方法能够保证所有资源处理权的唯一性。

证明:下面分情况证明。

1)节点加入。若WAOI内只有这一个节点,则WAOI内的所有共享资源都由该节点处理。若SAOI内有节点,EAOI内无论有无节点,由算法1的节点信息更新同步可知,当SAOI内的所有节点都改变了节点信息后,才重新进行资源的处理。当SAOI内无节点、EAOI内有节点时,节点将从邻居AOI获取SAOI内的共享资源,由算法2的邻居共享资源的恢复步骤可知,恢复过程仍然遵守处理权分配方法,即不允许SAOI内的资源被处理。因此,处理权在节点加入前后都是唯一的。

2)节点跨域。由算法3可知,节点将会通知新旧AOI的所有代理节点同步进行节点信息更新,更新完成后才进行资源的处理。若新AOI内没有节点,则需进行共享资源转换,转换结束后才进行资源的处理。若旧AOI内没有节点,则旧AOI的所有共享资源要重新分配给新AOI的所有代理节点,分配完后才进行资源的交互。因此,处理权在节点跨域前后也是唯一的。

3)节点失败。其他节点会自动删除失败节点,并重新计算属于失败节点的资源处理权。因此,处理权在节点失败前后都是唯一的。

以上表明资源处理权分配方法保证了资源处理权的唯一,从而为SAM模型保证所有资源状态的一致性提供了前提条件。证毕。

4.2 性能评价

下面从多个方面说明传统MMOG,SSC模型与SMA模型的性能差异。

响应速度:即从发出请求到接收到处理结果这一过程的时间差。由于代理节点之间是直接进行交互的,因此SMA模型的消息传输最长时间可以是 μ(一次单向传输时间);对于一般的触发响应事件,更新的节点自身信息直接发送给其他节点的时间也可以是 2μ 相对于传统MMOG和SSC模型的恒定速度 2μ 来说,SMA模型具备了较好的事件响应能力。

节点负载:事件的处理都被平均地分配给了所有的代理节点。同时管理节点维护代理节点链表和进行邻居节点发现的开销也相对较小。与SSC模型的slave节点相比,SMA模型会多出部分计算消耗,但与SSC模型的master节点相比性能就得到了极大的提高。

伸缩性:由于所有的事件处理都从服务器转移到了客户端节点,因此整个系统的存储能力和处理能力将得到极大的提高。相应地,游戏世界的资源和节点数量也会成倍地增加。这一点正好解决了传统MMOG中不断增大的资源需求同有限服务器负载能力之间的矛盾,且与SSC模型的master节点只能承受有限的slave节点相比,SMA模型也具有了更好的伸缩性。

健壮性:指系统稳定且持久运行的能力。节点变动时,SMA模型可在较短时间内恢复到正常通信状态,且影响范围也是局部的,因而能够保证系统稳定长久的运行。若是传统的MMOG架构,一旦服务器崩溃,则整个游戏便无法进行下去。当然SSC模型也有较好的健壮性,

综上所述,SMA模型在各个方面都具有明显的性能优势。

结束语 本文的贡献在于设计了一种高效的P2P MMOG模型。本模型由节点加入算法、邻居发现算法、节点跨域和失败算法构成其处理核心,并有效地将资源处理权均衡分配给了所有的节点,体现了P2P技术的优势。同时,在理论上证明了模型的工作机制能够保证所有资源状态的一致性。文中通过对模型响应速度、负载、伸缩性和健壮性的分析表明,本模型具备良好的性能。

SMA模型还需改善和补充的方面有:1)一个孤立AOI内的资源如何分配处理权。2)如何防止欺骗行为。问题1)指的是当一个WAOI内都没有节点加入时,SAOI内资源的行为交给谁来处理。例如一个NPC在周围很远范围内没有任何玩家的地方巡逻,NPC的行为由谁来维持。本文中的解决方案是由数据服务器来维持这些行为。这并不是好的解决方案,因为没有充分利用P2P网络的计算和存储能力。问题2)可以作为一个较大的研究方向。由于P2P分布式处理的特点,构建在其上的SMA模型也必然存在多种作弊方式,例如修改包、丢弃包、多节点合作等,因此安全性问题也是下一步工作的重点。

参考文献

[1] Hampel T, Thomas B, Robert H. A Peer-to-Peer Architecture for Massive Multiplayer Online Games[C]// Proc. of 5th ACM SIGCOMM Workshop on Network and System Support for Games. Singapore. ACM, 2006: 1-4

[2] Anthony Y, Vuong Son T. MOPAR: A Mobile Peer-to-Peer O-

verlay Architecture for Interest Management of Massively Multiplayer Online Games[C]// Proc. of the International Workshop on Network and Operating Systems Support for Digital Audio and Video. Washington, ACM, 2005: 99-104

[3] Christoph N, Nicolas P, Matteo V, et al. Challenges in Peer-to-Peer Gaming[J]. ACM SIGCOMM Computer Communication Review, 2007, 37(1): 79-82

[4] Zhou Jin, Tang Li, Li Kai, et al. A Low Latency Peer to Peer Approach for Massively Multiplayer Games[M]. Berlin Heidelberg: Springer Verlag, 2006: 120-131

[5] Baughman N E, Liberatore M, Levine B N. Cheat-proof playout for centralized and distributed online games[C] // Proc. of Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Anchorage: IEEE, 2001, 15(1): 104-113

[6] Chris G D, Daniel Z, Virginia L, et al. Low latency and cheat-proof event ordering for peer-to-peer games[C]// Proc. of the 14th International Workshop on Network and Operating Systems Support for Digital Audio and Video. Cork: ACM, June 2004: 134-139

[7] Corman A B, Douglas S, Schachte P, et al. A Secure Event Agreement (SEA) protocol for peer-to-peer games[C]// Proc. of the First International Conference on Availability, Reliability and Security. IEEE Computer Society, 2006: 34-41

[8] Chan M C, Hu Shun Y, Jiang J R. An Effcient and Secure Event Signature (EASES) Protocol for Peer-to-Peer Massively Multiplayer Online Games[J]. Journal of Computer and Telecommunications Networking, 2007, 52(9): 1037-1046

[9] Knutsson B, Lu Honghui, Xu Wei, et al. Peer-to-Peer Support for Massively Multiplayer Games[C] // Twenty-third Annual-Joint Conference of the IEEE Computer and Communications Societies. 2004: 96-107

[10] Iimura T, Hazeyama H, Kadobayashi Y. Zoned Federation of Game Servers: a Peer-to-Peer Approach to Scalable Multi-player Online Games[C] // Proc. of 3rd ACM SIGCOMM Workshop on Network and System Support for Games. ACM, 2004: 116-120

[11] Cecin F R, Martins M G. FreeMMG: a hybrid peer-to-peer and client-server model for massively multiplayer games[C] // Proc. of 3rd ACM SIGCOMM Workshop on Network and System Support for Games. Portland: ACM, 2004: 172-172

[12] Douglas S, Tanin E, Harwood A. Enabling massively multiplayer online gaming applications on a p2p architecture[C]// Proc. of the International Conference on Information and Automation. Colombo: IEEE, 2005: 7-12

[13] Rowstron A, Druschel P. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems[R]. LNCS-2218 In Penn CIS Tech Report. Berlin Heidelberg: Springer Verlag, 2004: 329-351

[14] 石祥滨，王越，李强. 一种适合 P2P MMOG 的移动代理迁移策略[J]. 计算机工程，2008，34(4)：146-148

[15] 石祥滨，杜玲，邢元胜. 基于 P2P 的 MMOG 中动态负载均衡算法[J]. 计算机工程，2007，33(16)：86-87，91

(上接第 98 页)

[2] Shah S. Dynamic Channel-aware Bandwidth Management in IEEE 802. 11 Networks [D]. Urbana- Champaign, USA: University of Illinois, September 2005

[3] Zhang Z J. Guaranteeing End-to-End Adaptive QoS in Wireless Multimedia Networks[J]. Chinese Journal of Computers, 2004, 27(8): 1064-1073

[4] Subramanian V, Kalyanaraman S, et al. Hybrid Packet FEC and Retransmission-based Erasure Recovery Mechanisms (HARQ) for Lossy Networks: Analysis and Design[C]// Proc. of Wireless Systems: Advanced Research and Development (WISARD). Bangalore, India (Invited paper), January 2007

[5] Azouzi R E, Peyre T, Benslimane A. Optimal design of hybrid FEC/ARQ schemes for real-time applications in wireless networks[C]// Proc. of the 2nd Workshop on Wireless Multimedia Networking and Performance Modeling (WMuNeP). Terromolinos, Spain, 2006: 11-18

[6] Mei Z. An Adaptive Forward Error Correction Algorithm for Streaming Video[J]. Journal of Software, 2004, 15(9): 1405-1412

[7] Barakat C, Fawal A A. Analysis of link-level hybrid FEC/ARQ for wireless links and long-lived TCP traffic [J]. Performance Evaluation (Elsevier), 2004, 57(4): 453-476

[8] Kim W, Jang H J, Kim G Y. Transmission Rate Prediction of VBR Motion Image Using the Kalman Filter[C] // Proc. of Workshop on Information Systems Information Technologies (ISIT). Seattle, Washington, USA, 2006: 106-113

[9] 陈云鹏，张培仁，高修峰. 无线视频的一种码率控制方法[J]. 通信学报，2006，27(3)：94-98

[10] Jing Z R, Yang Y, et al. Signal detection and estimation [M]. Beijing: Chemical Industry Press, 2004

[11] StarWarsIV[EB/OL]. www. tkn. tu-berlin. de/research/trace/ltvt. html

[12] Cai J F, Chen C W, Zhang Q. An FEC - based Error Control Scheme for Wireless MPEG-4 Video Transmission[C]// IEEE Wireless Communications and Networking Conference (WCNC 2000). Chicago, USA, 2000: 1243-1247

[13] Johanson M. Adaptive Forward Error Correction for Real-time Internet Video[C]// Proc. of the 13th Packet Video Workshop. Nantes, France, 2003

[14] RFC3580. IEEE 802. 1X Remote Authentication Dial in User Service (RADIUS) Usage Guidelines[S]. 2003