一种面向P2P空间查询的路由恢复方法

刘 丹 谢文君
(华中师范大学信息技术系 武汉 430079)

摘 要 基于一种P2P空间查询系统,分析了该系统中节点失效时可能出现的问题,提出了一种基于空间接管的路由恢复方法,以在节点失效时维持整个数据空间的完整性。同时给出了在这种路由恢复方法下的节点加入以及空间查询算法。测试表明,这种路由恢复方法能有效地解决节点失效带来的空间查询消息"回溯"、节点不能加入系统等问题,增强了系统的可用性。

关键词 P2P,分组,空间查询,路由恢复
中图法分类号 TP393 **文献标识码** A

P2P Spatial Querying-oriented Routing Recovery Method

LIU Dan XIE Wen-jun
(Department of Information Technology, Huazhong Normal University, Wuhan 430079, China)

Abstract This paper analyzed the triggered problems due to invalidation, and put foward a routing recovery method based on taken space, so as to maintain the integrity of whole data space when peers fail. At the mean time, algorithms of peer joining and space quering based on this routing recovery method were provided. Test results present that this routing recovery method can resolve the problems of back-track quering messages and joining failing effectively, and can reinforce the usability of the system.

Keywords P2P, Group, Spatial query, Routing recovery

1 引言

P2P系统在规模性、可用性、可扩展性等方面表现出来的优势使得如何在P2P环境下支持空间查询成为近年来的研究热点并出现了一些研究成果。由于P2P系统中的节点故障不可避免,因此,本文面向P2P空间查询,重点讨论在出现节点失效情况时的路由恢复机制,从而尽可能减小节点失效对P2P空间查询的影响。

P2P系统中在出现节点失效情况时的弹性表现在3个方面[1]:一是系统中数据的复制,二是系统的静态弹性,三是系统的路由恢复。其中静态弹性指的是在有节点失效并且路由表没有更新的情况下路由失败(两个网络中存在的节点不能通信)的比例。大多数结构化覆盖网络都是基于某种有结构图,比如Pastry[2],Tapestry[3]使用的是树结构;Chord[4],Chord#[5]使用的是环形结构;CAN[6]使用的是超立方体结构;Viceroy[7]和Ulysses[8]使用的是蝴蝶网结构;Kademlia[9]使用的是异或结构等等。文献[10]通过实验证明基于环形结构的覆盖网络由于具有更多的路由选择以及天然支持连续邻居而表现出更好的静态弹性。这意味着环形结构的覆盖网络在实际应用中可以以较低的频率来更新节点的路由表,从而降低网络开销。路由恢复指的是系统发现失效节点后自动更新其它未失效节点路由表从而尽可能恢复查询路由。

本文基于环形P2P结构Chord#,借鉴Chord以及CAN的路由恢复思想,提出一种面向P2P空间查询的路由恢复方法。

2 P2P空间查询系统的构造与节点组织

本文假定每个节点针对本地存储的数据维护了一个或多个本地空间数据边界矩形(Local SpatialData Mininum Bounding Rectangle,LSD-MBR)。系统中所有的节点依据功能划分为路由节点和数据节点,分别用Route_peer和Data_peer表示。其中Data_peer指的是仅负责根据本地数据计算查询的数据源节点,而Route_peer指的是从Data_peer中挑选出来的负责根据索引信息路由查询消息的数据源节点。另外,系统中每个数据源节点将LSD-MBR的中心作为自己的"代表点",其主要目的是通过代表点来决定节点在数据空间中的位置,从而决定在覆盖网络中的位置。系统的构造过程描述如下:

(1)第一个数据源节点加入,从功能角度上说,它是一个Route_peer。

(2)其他的数据源节点通过系统中任何一个已经存在的节点加入系统,并且它们都向Route_peer注册关于自己的地址、LSD-MBR、代表点等信息。

(3)当Route_peer发现维护的节点数量(包括自己)超过预先设置的某个门限值时(用Load_Max表示),对整个数据空间进行分裂,并从分裂出去的节点中选择一个节点作为

到稿日期:2012-02-04 返修日期:2012-06-01 本文受华中师范大学中央高校基本科研业务费项目(CCNU11A010)资助。
刘 丹(1978—),男,博士,讲师,主要研究方向为对等计算环境下的空间数据存储与查询、WebGIS,E-mail:rock_fall@tom.com。

Route_peer，并将该节点的地址告知所有分裂出去的节点。这样整个数据空间就一分为二，分别包含了若干数量的数据源节点。

(4)上述过程不断重复，随着数据源节点的不断加入，数据空间最终被分裂成若干个子空间。

利用层次化的环形结构(Chord#)来组织网络中的空间数据源节点。每当系统发生分裂产生新的子空间时，这个新的子空间的 Route_peer 就可以根据与 CAN 类似的 ID 分配方式得到一个 ID，并据此按照 Chord# 的节点加入协议加入到系统中。最终由各个子空间的 Route_peer 形成一个顶层的环状网络。为了保证查询路由的效率以及适应系统中节点的动态变化，需要为系统中的每个节点构造路由表，描述如下。每个 Route_peer 以递归的形式维护系统中另外 m 个 Route_peer 的地址信息(假设系统中有 N 个 Route_peer，则 $2^m=N$)。另外，还需要维护一定数据量的后继 Route_peer 地址信息和本组内所有 Data_peer 的地址列表 Data_peerlist，如表 1 所列。

表 1　Route_peer 路由表定义

符号	定义
finger[i]	finger[i−1], finger[i−1], (0<i≤m)
successor	finger[0]
predecessor	本节点的前一个 Route_peer
successorlist	后继 Route_peer 列表
Data_peerlist	Data_peer[j], (0≤j≤Load_Max−1)

上述系统能够较好地支持 P2P 空间点查询、范围查询、区域查询等。相关查询算法在文献[11]中有详细介绍，在此不再赘述。

3　Route_peer 失效问题分析

在本文的 P2P 空间查询系统下，节点失效分为两种情况：第一种是 Data_peer 失效，第二种是 Route_peer 失效。虽然 Data_peer 的失效使得系统的整体数据有所有损失，即在 Data_peer 失效期间不能贡献存储在本地并且符合空间查询条件的结果，但本文并不考虑第一种情况，因为这主要涉及到数据复制的问题。对于 Route_peer 的失效，主要会出现 3 个问题。第一个是本组内所有正常的 Data_peer 不能贡献存储在本地并且符合空间查询条件的结果。出现这类问题的原因是当查询到达目的地分组时，需要 Route_peer 将查询广播给或者有选择地转发给本组 Data_peer。第二个问题是系统中会出现空间查询消息“回溯”的情况，给网络造成不必要的开销。第三个问题是即使失效的节点经过一段时间恢复正常，也有可能不能重新回到系统中，并且新的节点也有可能不能加入。下面对空间查询消息“回溯”进行简要分析。

假设系统中的空间分裂状态如图 1 所示，并且分组 0111 的 Route_peer 失效，那么根据 Chord 的路由恢复算法，分组 1000 的 Route_peer 将取代分组 0111 成为分组 0110 的后继节点。如果现在分组 0100 内的节点提出一个点查询(0.3, 0.8)，即属于分组 0111 所对应的子空间，那么根据文献[11]的点查询路由算法，可以得出查询消息的转发过程：首先消息被转发给分组 0110，然后转发给分组 0111。由于这时分组 0111 的 Route_peer 失效，并且 1000 是第一个大于 0111 的分组 *ID*，因此分组 1000 收到了这个消息。分组 1000 通过解析点坐标，将消息转发给分组 0000。分组 0000 又将消息转发给 0100，接着是 0110，最后又转发给了分组 1000，这一过程将无限循环下去，如图 2 所示。

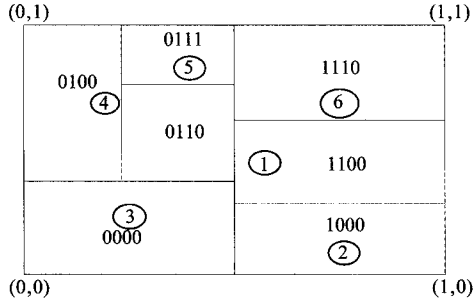
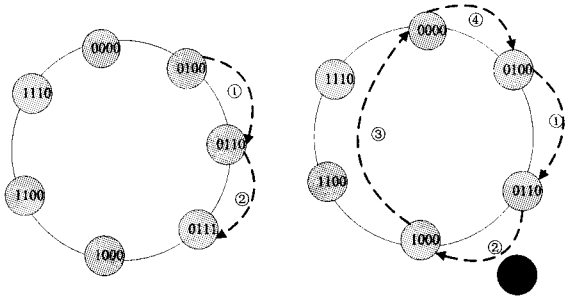


图 1　空间分裂历史示意图



正常情况下的查询消息路由　　节点0111失效情况下的查询消息路由

图 2　空间查询消息“回溯”示意图

新的节点无法加入系统的原因与查询消息“回溯”相同，本质上都是因为随着 Route_peer 失效，由其管理的分组所对应的子空间也不存在了。因此，路由恢复的关键在于不仅要保证 Chord# 环的完整性，还要保证整个数据空间的完整性。

4　基于空间接管的路由恢复算法

基于子空间接管的路由恢复算法的关键就是当系统中的某个 Route_peer 发现自己的后继节点失效后如何处理。基本思路为：系统中的每个 Route_peer 不仅维护 r 个后继节点的地址，而且还维护它们所在分组对应的子空间范围。当节点 p 发现后继节点失效后，首先将失效节点的后继节点作为自己新的后继节点，然后将失效节点所管理的子空间发送给新的后继节点，即由新的后继节点来接管失效节点。

这里需要说明的是，在 Chord 的路由恢复处理中，当完成了替换后继节点的工作后，便可以删除关于失效节点的相关信息。但在本文的算法中则需要继续保留关于失效节点的相关信息，因为新的后继节点可能依然是失效节点，如果将旧的失效节点的相关信息删除，会造成子空间信息的彻底丢失。如果新的后继节点在系统中正常运行并且收到了失效节点管理的子空间信息，则将自己原先管理的和接管的子空间信息合在一起作为 p 的第一个后继节点所管理的子空间信息发送给 p，这时 p 删除失效节点的相关信息。如果失效节点的后继节点也失效了，则重复上述过程直到找到正常运行的后继节点或超过 p 的后继节点的列表范围。为了方便后续描述，在此对本文提出的路由恢复算法所涉及到的一些数据结构进行定义，如表 2 所列。

表 2　路由恢复算法所涉及到的数据结构定义

符号	定义
succList[i]	所有后继 Route_peer 信息集合，1≤i≤r，r 为预先定义的所维护的后继 Route_peer 个数
succList[i]. address	后继 Route_peer 地址信息
succList[i]. Area	后继 Route_peer 所管理的子空间范围信息（包括可能接管的子空间范围信息）
takenGIDSet[j]	接管的分组 ID 集合 0≤j≤m，m 为接管的分组个数
takenAreaSet[j]	接管的分组子空间范围信息集合 0≤j≤m，m 为接管的分组个数
takenSplitHisSet[j]	接管的分组分裂历史信息集合 0≤j≤m，m 为接管的分组个数

在算法 1 中，takeAreanum 的初始值为 0（并不是每次调用路由恢复算法时都将其置 0，而是在收到了后继节点成功接管的确认消息后置 0），用来计算需要接管的失效节点个数。temptakeArea 的初始值为空，用来存储当前需要发送给后继节点接管的子空间信息。

算法 1　routing recovery

```
//takeAreanum 的初始值为 0，temptakeArea 的初始值为空
1. if (finger[0] failed &&takeAreanum<succList. lenght)
2.    takeAreanum++
3.    finger[0]=succList[takeAreanum]
4.    for i=0 to takeAreanum-1 do
5.        temptakeArea. add(succList[i]. Area)
6.    end for
7.    forword temptakeArea to finger[0]
8. end if
```

5　路由恢复下的节点加入与 P2P 空间查询

本小节分别介绍在给出的路由恢复方法下，如何进行节点加入以及空间查询。

（1）路由恢复下的节点加入

节点加入算法思路为：当某个 Route_peer 收到加入请求后（表现为一个点坐标和一个分组 ID），首先查看请求中包含的分组 ID 是否等于自己所在分组的 ID，如果等于，则调用点查询算法分析自己所在分组是否就是目的地分组，如果是则接管提出请求的节点，如果不等于，则将新的加入请求（将原来的分组 ID 更新为重新计算过的分组 ID）转发出去。如果请求中包含的分组 ID 不等于自己所在分组的 ID（即在自己与前继之间），那么说明有失效情况发生。遍历自己接管的分组集合，将请求交给其 ID 等于请求分组 ID 的分组并重复上述过程（不同之处在于分析过程在接管节点本地进行），直到最后加入到某个分组。需要说明的是，如果一个失效后恢复的 Route_peer 在接管分组集合中找到了目的地分组（即原本属于它管理的分组），则将这个分组和接管分组集合中比其 *ID* 小的分组一起接管。

举例来说，如图 3 并参照图 1 所示，假设节点 0100，0110 和 0111 同时失效并由节点 1000 接管（为了描述方便，用分组 *ID* 来表示 Route_peer）。当一个代表点在分组 0110 子空间范围内的 Route_peer 希望通过节点 1100 加入系统时，节点 1100 通过分析代表点坐标和本分组的分裂史，将加入请求转发给节点 0000。节点 0000 通过分析将请求转发给 0100。由于 0100 失效，因此请求被节点 1000 收到。节点 1000 在自己的接管分组集合中找到了 0100，因此通过 0100 的分裂史继续解析请求。由于代表点坐标在解析出的结果 0110 范围内，因此节点 1000 将分组 0100 和 0110 的相关信息一起转发给请求加入的 Route_peer（即由这个 Route 管理分组 0100 和 0110），并继续接管分组 0111 。

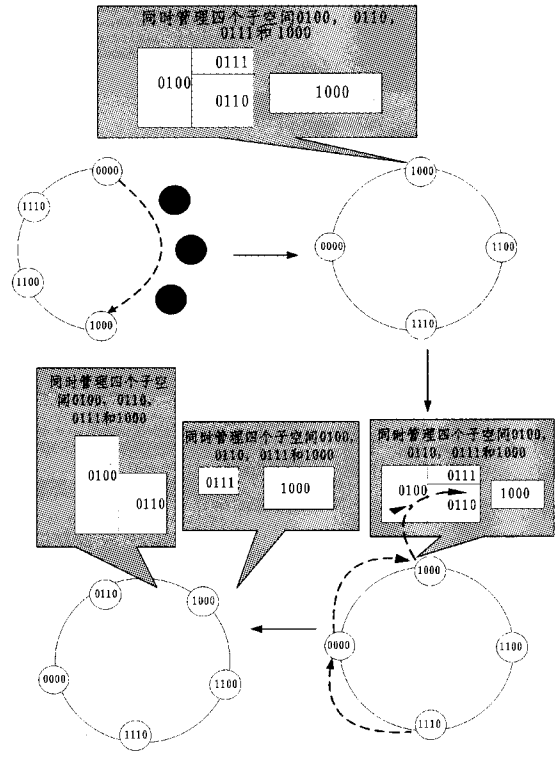


图 3　Route_peer 失效状态下的节点加入示意图

（2）路由恢复下的 P2P 空间查询

路由恢复下的几种 P2P 空间查询思路基本相同，本文以区域查询为例进行介绍。如算法 2 所示，当 Route_peer 收到某个由它负责的区域查询的时候，首先比对该查询的目的地分组 ID 和 LocalGID，如果相等则执行 AreaQuery 方法，如果不相等，则说明存在被接管分组（因此目的地分组 ID 落入了 LocalGID 和前继分组 ID 之间）。然后遍历 takenAreaSet 中所有被接管分组的子空间范围，如果和区域查询范围 qrange 相交，则在相应的分组中执行区域查询。接着判断区域查询范围是否与本分组的子空间 LocalArea 相交。最后执行 filterArea 方法，该方法返回的是在本地 KDB 树中与 qrange 相交的本地子空间结点的右兄弟子空间（执行该方法的目的是防止查询消息在系统中出现回溯的情况），并将新的区域查询转发给该子空间所属的分组。

算法 2　takenArenAreaQuery(qrange)

```
//当 Route_peer 收到某个由它负责的区域查询的时候
1. if 如果该查询的目上的地分组 ID 等于 LocalGID
2.    perform AreaQuery(qrange)
3. else
4.    if (isintersect(qrange,LocalArea))
5.        perform query
6.    end if
7.    for i=0 to takeInfoSet. length
8.      if (isintersect(qrange,takeInfoSet. AreaSet[i]))
9.          perfor query in takeInfoSet. AreaSet[i]
10.       end if
11.     end for
```

```
12.     DesArea=filterArea(qrange)
13.   if (DesAresa! =null &&
          ! DesArea∈takeInfoSet. AreaSet)
14.       forward AreaQuery(intersected(DesArea,qrange)) to De-
            sArea
15.   end if
16. end if
```

6 实验与分析

本文所有的仿真实验都是基于 P2P 仿真平台 PlanetSim 以及 Java SDK 1.6,实验程序运行环境为 Intel P4 3.0G,内存 512M。

第一个测试为随机生成由 2^{10} 个 Route_peer 组成的覆盖网络,后继节点列表大小为 10。然后分别随机使其中 10%,20%,30%,40%,50%的 Route_peer 失效并更新路由表,最后在不同的情况下每次随机加入 200 个节点(即随机选取代表点)。加入原则为如果代表点坐标落入未失效节点所在分组的子空间,则以 Data_peer 的角色加入,如果落入未失效节点接管的分组的子空间,则以 Route_peer 的角色加入。测试结果如图 4 所示,在未采用本章提出的路由恢复算法的情况下,加入成功率随着失效 Route_peer 的增加成比例减少。这是由那些代表点落入失效 Route_peer 所在分组的子空间的节点造成的。而在采用了路由恢复算法的情况下,直到节点失效比例超过 40%以后才出现加入失败。加入失败的原因是当节点失效比例超过 40%以后 Chord# 环出现了"断裂"现象,从而无法保证整个数据空间的完整性。

第二个测试为随机生成由 2^{10} 个 Route_peer 组成的覆盖网络,后继节点列表大小为 10。然后分别随机使其中 10%,20%,30%,40%,50%的 Route_peer 失效并且不进行任何路由表更新,最后在不同的情况下每次随机加入 200 个节点(即随机选取代表点)。加入原则与测试一相同。测试结果如图 5 所示,随着失效 Route_peer 比率的增加,节点加入成功率逐渐降低,这是由 Chord# 环"断裂"以及空间丢失逐渐增多造成的。另外,由于空间接管是基于路由表的更新来发现失效节点并最终确定新的后继节点(即接管节点),因此在未进行任何路由表更新的情况下,是否采用本文提出的路由恢复算法对加入成功率没有任何影响。

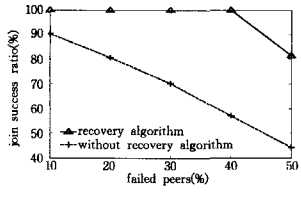


图 4 节点加入最佳成功率比较图

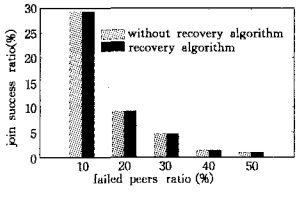


图 5 节点加入最差成功率比较图

第三个测试为随机生成由 2^{13} 个节点组成的覆盖网络,分组大小为 8,后继节点列表大小为 10。然后分别随机使其中 10%,20%,30%,40%,50%的 Route_peer 失效并更新路由表,最后在不同的情况下每次发送 100 个点查询消息(即随机选取查询点)。测试目标为系统中无效查询消息(即陷入"回溯状态"的查询消息)的百分比。测试结果如图 6 所示,在未采用本章提出的路由恢复算法的情况下,无效消息百分比随着失效节点的增加成比例增加。但当失效节点比例达到 60%时,无效消息百分比却减少了。这也是由于 Chord# 环出现"断裂"造成的。在 Chord# 环"断裂"的情况下,查询消息无法被顺利地转发,这反而降低了无效消息百分比。而在采用了路由恢复算法的情况下无效消息百分比一直为 0。

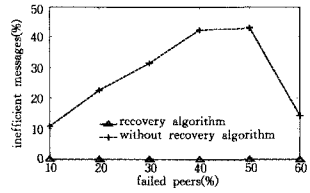


图 6 无效查询消息比率比较图

第四个测试为随机生成由 2^{13} 个节点组成的覆盖网络,分组大小为 8,后继节点列表大小为 10,随机失效系统中 10%的 Route_peer。然后分别在系统运行 50,100,150,200,250"步"情况下,发送 50 个查询范围为整个数据空间 1%的随机区域查询。测试目标为系统中点查询失败的百分比。计算方式为:1—实际响应了查询的节点总数/(应该响应查询的节点总数—失效节点总数)。需要说明的是,在 PlanetSim 中的"1 步"定义为:仿真器将系统中所有节点输出消息队列中的消息移至输入消息队列,并且在每个节点上调用相应的处理方法处理完输入消息。在本测试中,Route_peer 路由表项更新程序设置为每 5 步运行一次,Data_peer 每 20 步探测本组 Route_peer 一次。

测试结果如图 7 所示,在未采用本章提出的路由恢复算法的情况下,点查询失败百分比随着运行步数的增加而减少,当系统运行了 200 步后不再发生变化。这是因为随着运行步数的增加,Chord# 环的逐渐恢复增加了查询路由成功的可能性。运行到一定的时候,Chord# 环完全恢复但数据空间无法恢复,因此结果不再发生变化。在采用了路由恢复算法的情况下点查询失败百分比随着运行步数的增加而逐渐减少直至为 0,其原因可以从前面的分析中得出。另外,从图 7 可以看出,两种情况下的查询失败比率在系统运行步数较少时差别并不大,随着运行步数的增加,差距逐渐扩大。这是因为本章提出的路由恢复算法是建立在 Chord# 环恢复的基础上,而在系统运行的初始阶段,节点路由表更新不够充分,使得路由恢复算法的优势并不明显。

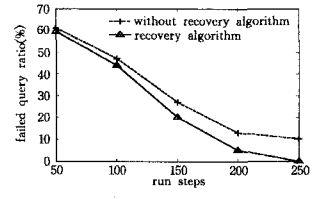


图 7 范围查询失败比率比较图

结束语 本文基于一种分组 Chord# 下的 P2P 空间查询系统,分析了该系统中出现节点失效时可能出现的问题,提出了相应的路由恢复方法。测试表明,这种路由恢复方法能有效解决节点失效带来的空间查询消息"回溯"、节点不能加入系统等问题,增强了系统的可用性。但在未进行任何路由表更新的情况下,是否采用本文提出的路由恢复算法对加入成功率没有任何影响。

由于本文主要从路由恢复角度处理节点失效,并未处理在 Data_peer 失效期间不能贡献存储在本地并且符合空间查

(下转第 59 页)

在的问题，然后着重分析了多速率多播的特点及优势。在此基础上，将资源分配算法与多速率及多播传输结合起来，提出一种新的具有多速率及多播传输能力的 Ad hoc 网络资源分配算法 MMRA。首先，给出了基于价格的多速率资源分配算法，它通过效用最大化框架来计算速率分配向量。接着，在资源分配算法上进一步扩展了多速率和多播传输能力。仿真结果表明：MMRA 算法具有良好的收敛性能，能够提高多播组的平均吞吐量和链路利用率，其性能优于另外两个算法 Alt-Bid 和 SPF，当网络负载增加时，MMRA 的优势更加明显，表现出了较好的稳定性和网络性能。

参 考 文 献

[1] Kuo W-H, Liu T, Liao W. Adaptive Resource Allocation for Layer-Encoded IPTV Multicasting in IEEE 802. 16 Wireless Networks [J]. IEEE Trans. on Multimedia, 2011, 13(1): 116-124

[2] Hua S, Guo Y, Liu Y, et al. Scalable Video Multicast in Hybrid 3G/Ad-Hoc Networks[J]. IEEE Trans. on Multimedia, 2011, 13(2): 402-413

[3] Shakkottai S, Liu X, Srikant R. The Multicast Capacity of Large Multihop Wireless Networks[J]. IEEE Trans. on Networking, 2010, 18(6): 1691-1700

[4] Alay O, Korakis T, et al. Dynamic Rate and FEC Adaptation for Video Multicast in Multi-rate Wireless Networks [J]. Mobile Networks & Applications, 2010, 15(3): 425-434

[5] Deb S, Srikant R. Congestion control for fair resource allocation in networks with multicast flows[J]. IEEE/ACM Transactions on Networking, 2004, 12(2): 274-285

[6] Kar K, Tassiulas L. Layered multicast rate control based on lagrangian relaxation and dynamic programming[J]. IEEE Journal on Selected Areas in Communications, 2006, 24(8): 1464-1474

[7] Li Ying, Tian Chao, et al. Network resource allocation for competing multiple description transmissions[J]. IEEE Transactions on Communications, 2010, 58(5): 1493-1504

[8] Lee H-W, Cho J W, Chong Song. Distributed max-min flow control for multi-rate overlay multicast [J]. Computer Networks, 2010, 54(11): 1727-1738

[9] Kao Y F, Huang J H. Price-based resource allocation for wireless Ad hoc networks with multi-rate capability and energy constraints [J]. Computer Communications, 2008, 31: 3613-3624

[10] 韩冰青，张宏，刘凤玉，等. 无线 Ad hoc 网络中 QoS 感知的跨层资源分配算法[J]. 软件学报，2010，21(12)：3138-3150

[11] Cui Y, Xue Y, Nahrstedt K. Optimal resource allocation in overlay multicast[J]. IEEE Trans. Parallel Distrib. Syst., 2006, 17(8): 808-823

[12] Jiang L, Walrand J. A distributed CSMA algorithm for throughput and utility maximization in wireless networks [J]. IEEE/ACM Transactions on Networking, 2010, 18(3): 960-972

[13] Boyd S, Vandenberghe L. Convex Optimization [M]. Cambridge University Press, 2004

[14] Curescu C, Nadjm-Tehrani S. A bidding algorithm for optimized utility-based resource allocation in Ad hoc networks[J]. IEEE/ACM Transactions on Mobile Computing, 2008, 7(12): 1397-1414

(上接第 50 页)

询条件的结果的情况，因此下一步的工作主要考虑如何从数据复制的角度应对节点失效。

参 考 文 献

[1] Gummadi K, Gummadi R, Gribble S, et al. The Impact of DHT Routing Geometry on Resilience and Proximity[C]// Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications. Karlsruhe, Germany, 2003: 381-394

[2] Antony I T, Rowstron, Rowstron P D. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems[C]// Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms. Heidelberg, Germany, 2001: 329-350

[3] Zhao B Y, Ling Huang, Stribling J, et al. Tapestry: A Resilient Global-Scale Overlay for Service Deployment[J]. IEEE Journal on Selected Areas in Communications, 2004, 22(1): 41-53

[4] Stoica I, Morris R, Karger D, et al. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Application[C]// Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. San Diego, California, United States, 2001: 149-160

[5] Schutt T, Schintke F, Reinefeld A, et al. Structured Overlay without Consistent Hashing: Empirical Results[C]// Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid Workshops (CCGRIDW'06). Singapore, 2006: 8

[6] Ratnasamy S, Francis P, Handley M, et al. A Scalable Content-Addressable Network[C]// Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. San Diego, California, United States, 2001: 161-172

[7] Malkhi D, Naor M, Ratajczak D. Viceroy: A Scalable and Dynamic Emulation of the Butterfly[C]// Proceedings of the 21st ACM Symposium on Principles of Distributed Computing (PODC '02). Monterey, California, United States, 2002: 183-192

[8] Kumar A, Merugu S, Xu J, et al. Ulysses: A Robust, Low-Diameter, Low- Latency Peer-to-Peer Network[C]// Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP '2003). 2003: 258-267

[9] Maymounkov P, Mazieres D. Kademlia: A Peer-to-peer Information System Based on the XOR Metric[C]// 1st International Workshop on Peer-to-Peer Systems (IPTPS '02). 2002: 53-65

[10] Gummadi K, Gummadi R, Gribble S, et al. The Impact of DHT Routing Geometry on Resilience and Proximity[C]// Proceedings of the 2003 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. Karlsruhe, Germany, 2003: 381-394

[11] 刘丹，谢文君. 一种基于 P2P 的空间数据索引方法[J]. 计算机科学，2012，39(8)：186-190