

CT-Cycloid: 一个基于 Cycloid 的 抗 Churn 的 P2P 系统

陈 敬, 禹继国

(曲阜师范大学 计算机科学学院, 山东 日照 276826)

摘 要: Cycloid 是一种常数度结构化 P2P 覆盖网, 它具有高扩展性、自组织、自适应开销低等优点, 然而像所有其它的常数度 P2P 覆盖网一样, 它不能很好地适应高 churn 环境。为解决这一问题, 在一个简单而新奇的设想的启发下, 基于生存期的策略和角色划分机制被应用来对 Cycloid 的设计进行改造, 构造了一个基于 Cycloid 的具有高抗 churn 能力的结构化 P2P 系统——CT-Cycloid。在 CT-Cycloid 系统中进行路由和定位的开销是 $O(\log S)$, 远低于一般系统中的 $O(\log N)$ ($S = N/\log N$)。大多数节点到来和离开时仅仅需要发送一条一跳到达的信息, 节点的失效会很快被检测到, 网络很快得以恢复稳定。理论分析和仿真实验都证明了 CT-Cycloid 与 Cycloid 相比查询路径更短, 适应高 churn 环境的能力更强。

关键词: P2P 覆盖网; Cycloid 覆盖网; churn

中图分类号: TP393

文献标识码: A

文章编号: 1673-629X(2010)07-0244-06

CT-Cycloid: a Novel Churn-Tolerant P2P System Based on Cycloid

CHEN Jing, YU Ji-guo

(Department of Computer, Qufu Normal University, Rizhao 276826, China)

Abstract: As an excellent scheme of constant-degree P2P overlay networks, Cycloid has high scalability, and pays low cost for self-organization and self-adaptation. Similar to other constant-degree P2P overlay networks, Cycloid cannot work very well in high-churn-level environments. To overcome this, modify Cycloid and propose CT-Cycloid, a novel churn-tolerant P2P system based on Cycloid, by applying a simple and novel scheme, a lifespan-based strategy and a role-division mechanism. The overhead of the novel system for routing and localization is $O(\log S)$, which is significantly lower than $O(\log N)$ in common P2P networks, where $S = N/\log N$, and N is the number of nodes in systems. Most nodes in CT-Cycloid just need to send a one-hop message when join or leave the overlay network. Nodes failure can be found quickly, and can be dealt reasonably. Both theoretical analysis and simulation have proved that CT-Cycloid needs smaller number of hops in searching and has higher churn-tolerant capability than Cycloid.

Key words: P2P overlay network; Cycloid; churn

0 引 言

对等网络, 亦称 P2P (Peer-to-Peer network) 覆盖网络, 采用对等模式工作, 能够充分利用网络带宽, 开发每个节点能力, 具有很高可扩展性^[1-3]。churn 是指由覆盖网参与者的匿名、自由性以及规模大的特点导致的大量节点频繁自发地加入、离开或失效的现

象^[4,5]。churn 问题是 P2P 网络发展必须要克服的难题之一。

Cycloid^[6,7] 是常数度的结构化对等网络。除具有一般 DHT 覆盖网的一切优点外, 它最大的优点是网络中每一个节点有常数度, 因此具有高可扩展性。同时由于每个节点的路由表规模固定且较小, 系统在自组织和自适应的开销也少。然而像所有结构化 P2P 覆盖网一样, 它难以应对网络中的高 churn 问题。

笔者对 Cycloid 做出调整, 并设计了相应的算法, 得到一个新的覆盖网结构 CT-Cycloid。它在节点正常离开时的花费为 $O(1)$, 且能够很好地应对节点的失效。另外, 还采取了相应措施来避免组长节点 (CT-Cycloid 结构中一个角色) 的频繁加入和离开, 从而

收稿日期: 2009-11-12; 修回日期: 2010-02-25

基金项目: 国家自然科学基金 (10471078); 山东省中青年科学家奖励基金 (2005BS01016); 山东省科技攻关计划 (2009GG10001014); 山东省教育厅科研项目 (J07WH05)

作者简介: 陈 敬 (1984-), 女, 硕士研究生, 研究方向为对等计算和计算机网络与通信; 禹继国, 博士, 教授, 硕士研究生导师, 研究方向为对等计算、无线网络、优化理论与算法设计等。

进一步减少了 churn 对系统稳定性的破坏。

1 相关工作

(1)Cycloid。

Cycloid 是一个常数度结构化 P2P 系统,它的拓扑结构模仿了立方体互连圈(CCC, Cube Connected Cycle)。一个 d 维 CCC 图是在一个 d 维超立方上将所有顶点替换成含 d 个节点的环而构成的图。图中包含 $d \times 2^d$ 个节点,每个节点的度数为 3。每个节点都有一个标识 $(k, a_{d-1}a_{d-2}\cdots a_0)$, 其中 k 是环上标识(cyclic index), 取值范围为 0 到 $d-1$ 之间的整数; $a_{d-1}a_{d-2}\cdots a_0$ 为立方体标识(cubic index), 取值范围为 0 到 2^d-1 之间的整数。

在一个 Cycloid 系统中,每个节点维护一个路由表和一个叶集,共 7 项,来维持它到系统其它节点之间的通信。

Cycloid 的路由算法模拟 CCC 的路由算法,分三个阶段(具体见参考文献[6,7])。算法最关键的思想与 Pastry 的定位、路由思想一致,就是不断地减小与目标节点之间的距离。每一阶段都是 $O(d)$ 步,所以总的路径长度就是 $O(d)$ 。

(2)基于生命周期的机制。

Stutzbach 和 Rejaie 通过对三个 P2P 应用(Gnutella、Bit Torrent 和 Kad)的研究来探究覆盖网的 churn 特性^[8]。得出对等体的已在线时间它是它剩余在线时间的一个很好的预测。这一结论启发了一种新的覆盖网 churn 应对策略——基于生命周期的组织策略。

Bustamante 和 Qiao 最先通过让节点选择新邻居时考虑最老的节点的方法来应用这一策略^[9]。他们的实验结果证明了这种策略在增加系统的稳定性方面非常有效。后来,他们又将这种策略应用到最近被广泛使用的查询相关的方法上,包括查询散布、缓存和复制^[10]。结果显示,基于生命周期的策略和这些增强机制的联合在很大程度上增强了系统的稳定性。

我们将基于生命周期的策略应用到 CT-Cycloid 的设计中——在肩负重任的组长节点离开时,被选作新组长的节点是组中现存最老的节点。因为根据以上研究结果这些节点趋向于在系统中生存更长时间。因而避免了组长频繁地加入和离开给系统带来的 churn。

(3)一个简单的抗 churn 机制。

Pandurangan 和 Jagannathan 提出了一个将一个静态网络转变为具有高抗 churn 能力的动态 P2P 覆盖网的方法^[11]。其过程是让 $O(\log N)$ 个动态网 G 中的节点占据静态网络 H 中的一个顶点(其中 N 为覆盖网 G 的大小),也就是让这些节点拥有相同的 ID,负责所有

被映射到该节点的数据对象。他们从数学上证明了该方案的有效性,但并没有提出具体实施方案。另外,他们建议网络中所有的节点拥有相同的角色和相同的责任,因而也就拥有相同规模的路由表($O(\Delta \log N)$),其中 Δ 为静态网络的最大节点度数。然而,因为有 $O(\log N)$ 个节点可以为同一个查询服务,所以有必要让一个节点充当调节者,于是进一步提出角色划分机制,并详细设计了覆盖网工作算法。

2 CT-Cycloid

2.1 CT-Cycloid 的系统结构

将一个 d 维 CCC 图的每一个顶点扩展为一个规模为 $O(\log N)$ 的节点组(其中 N 是将要构造的覆盖网的大小),就得到了一个 d 维 CT-Cycloid 的拓扑。如图 1 所示是使用一个静态的 3-维 CCC 图构造一个 CT-Cycloid 的拓扑示意图。CT-Cycloid 的每一个组由一个组长和若干个普通成员组成,组中所有成员共有一个 ID(即其组标识符),每个普通成员节点除了一个组标识外还有一个成员标识符。如在这个 3-维 CT-Cycloid 中,组(2,010)中有 8 个组员节点,它们的共同 ID 为(2,010),成员节点(3,2,010)的成员号为 3。

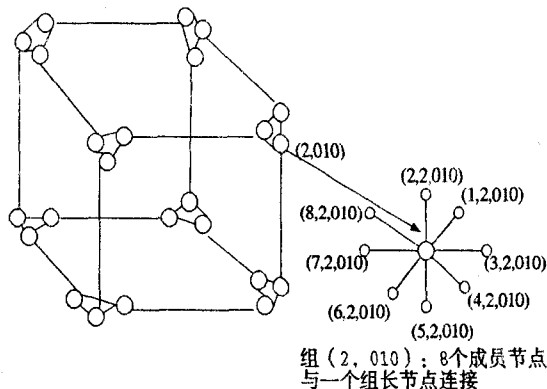


图1 CT-Cycloid 的拓扑图示

在路由过程中,普通成员对外组节点是透明的,并且同组成员之间也是透明的。仅组长节点对于自己的成员和其邻组组长节点来说是可见的。组长节点负责分配组成员号、给组成员安排任务、检测组成员的存活情况、离开覆盖网时任命新组长等工作。组中的普通成员则负责完成组长分配的任务(向新组员发送本组负责的数据对象、处理查询请求等)以及监视组长的存活状态等。

2.2 路由表的组织方式和路由策略

CT-Cycloid 继承了 Cycloid 的路由表的组织方式和路由策略,并做出一些扩展。

路由表的组织方式如下:组长节点中除了从 Cy-

cloid 继承的项以外,还保存一个成员表(如表 1)。其中包括所有成员的成员号、IP 地址、值日情况以及超时值。普通成员的路由表包括从 Cycloid 继承来的 7 项和组长的地址,共 8 项。

表 1 组长节点保存的成员表

成员号	IP 地址	值日	超时值
2	...	false	5
3	...	false	2
5	...	true	2
7	...	false	8

2.3 节点标识符的获取

节点 $(k, a_{d-1}, a_{d-2}, \dots, a_0)$ 的环标识 k 的获取方法是从 1 到 d 之间的整数中随机选择一个,立方体标识每一位的获取方法则是从 0 和 1 之间随机选择。这样可以把节点均匀地映射到整个标识符空间,从而有利于实现负载均衡。

2.4 新节点加入算法

新节点加入算法如下:

```
// 节点 n 通过对等体 G 加入覆盖网
n.join(G)
X = G.CycloidRouteTo(N);
if(X.ID == N.ID)
    Y = X.whosOnDuty();
    if(Y != null)
        Y.init(N);
    else
        X.init(N);
else
    N.setupAGroup(X);
```

算法 1:新节点加入算法

新节点 n 使用前述的方法计算出自己的标识 ID,并通过某种方式认识到覆盖网中一个现存对等体 G 后将执行算法 1。它首先通过 G 以自己的 ID 为目标向自己所属组发送一条消息。如果它所属的组已经存在,该信息将会被路由到组长 X 。收到该信息后组长节点会从成员表中找出值日生节点 Y ,并将信息转发给 Y ,然后由 Y 来对新节点进行初始化,即将本组保存的文件数据,8-项路由表,以及组长节点赋给新节点 n 的成员标识符号发送给 n 。

当然,如果组中仅有一个组长节点那么所有的工作将由组长自己完成。成员标识符与节点在组中已经存活的时间成反比,由节点到达该组的时间决定,到达越早得到的标识符越小。为此,组长节点维护一个随新成员到达而递增的变量 k 。

如果信息到达的节点 X 不是 n 所属组的组长节点,则意味着该组还不存在,那么 n 将执行 Cycloid 的

新节点加入算法,建立自己的组并成为组长。

2.5 数据插入

为插入一个文件,节点首先使用 SHA-1 计算出文件的标识符,然后将该文件发送到标识符与文件标识符相同或最接近的组中。收到信息的组长节点会记录该文件信息,并将该信息通过当前的值日节点转发到所有的成员节点。

2.6 对象查询方法

查询信息被使用 Cycloid 路由算法发送到对象 O 所在组的组长节点 X ,然后 X 查找到当前的值日节点 Y ,最后由 Y 对查询进行回复。

2.7 节点的离开

```
n.depart()
if(N.isLeader())
    X = theOldestMem();
    handover(X, memList, routingTable);
    newLeaderNotice(X);
else
    notifyMyLeader(leader, N);
```

算法 2:节点离开算法

如算法 2 所示,如果要离开的节点 n 是一个组长节点,离开前它将完成下面的工作:选择组长成员号最小(即组中最老的)的节点 X 作为新的组长,并向 X 发送一条包括本组成员列表和路由表的信息,完成职务的移交工作。然后向路由表中的节点,包括其它组的组长节点和本组的成员节点发送一条信息,告诉它们新组长节点的信息,以使它们进行路由表的更新。

如果离开的节点 n 是普通的组成员节点,该节点仅仅需要向自己的组长发送一条自己离开的信息,收到该信息后组长节点将 n 的信息从成员列表中删除。当然,节点 n 离开前不发送任何信息也没有关系,因为组长节点在分配任务或进行周期探测时会很快发现它的离开。

2.8 节点失效离开

正如刚刚所提到的,组长会通过发送 ping 信息对成员节点的存活情况进行周期的探测。如果组长节点在一定的时间间隔内没有接收到回复信息,它就会判断被探测的节点已经离开,并且删除其相关记录。每个组长节点周期进行的探测过程如算法 3。

关于何时选择哪一个节点如何进行探测,有下面的规定:

a. 如果组长节点没有除探测以外的工作,它就执行周期探测工作(算法 15-20 行)。如果在探测之前发现节点超时,那么它会删除节点的成员信息。

b. 当组长节点忙于其它工作如处理文件请求时,

探测工作会被停止(算法 3、4 行)。这种情况下,仍有方法保证成员表中的节点是存活的。收到任务的节点需要回复一条信息,告诉组长自己还存活并会完成任务。如果没有收到回复,组长会将任务交给下一个成员,并对没有发回复的节点进行探测,看其是否真的已经“死亡”。

```
//每个组长节点周期进行以下操作
1  do fever
2  //组长正忙于探测以外的工作
3  if (busy)
4      return 1;
5  else
6      //探测因组长忙于其它工作而被中止过
7      if(paused)
8          //值日节点超时
9          if(m=nextOnduty() is timeOut)
10             delete(m);
11             m=null;
12         else
13             ping(m);
14             return 1;//完成本次探测
15 //正常的周期探测过程
16 while(m=next() is timeOut)
17     delete(m);
18 if(m is not null)
19     ping(m);
20     return 1;//完成一次探测
21 return 0;//成员表为空
```

算法 3:组长节点的周期探测算法

c. 当组长在探测被中止后开始一次探测时,首先被探测的节点是下一个将要执行任务的节点(算法 7-14 行)。这样做可以保证下一次任务被交给存活节点,保证了工作效率。

很明显,由于这些策略的使用,普通节点无通知的离开可以很快被检测到。事实上,组长节点的失效离开也可以被很快发现。如果组长节点失效离开,它的组员们会因在一定时间内没有收到任务也没有被探测而判定组长节点已经失效。这种判定可以被保证是正确的,否则就出现了矛盾——如果组长节点没有死亡且组员节点没有收到探测信息,则说明组长正忙于其它工作,但是此时节点却没有收到任务。

使用这样的方法做出组长失效的判断后,组成员节点开始使用下面的方法竞选组长。所有的组员节点发送一条信息到自己的左内叶节点(如果左内叶节点不存在,选择右内叶节点。如果内叶节点都不存在则选外叶节点),告诉该节点本组的组长已经“死亡”,并

要求它协助新组长竞选工作。

为完成这一目标,信息中将包括该组的 ID、节点自己的 IP 地址和成员号。使用收到的成员号和对应的 IP 地址,该邻居节点会为该组构建成员表,并选成员号最小的那个节点作为新组长。同时该邻居还会依据自己的选择更新自己的邻居表。然后该邻居会将新建立的成员列表通过一条信息发送给新组长节点,并向所有其它成员节点发一个关于新组长的通知。节点并不知道何时会被选作“新组长竞选”的主持人,所以每个组长节点时刻对收到的信息进行判断,如果是这类信息,该组长节点将执行刚刚描述的工作,如算法 4。

```
do forever
msg = receives( * );
if (msg 是竞选帮助请求)
    memList = buildMemList(msg);
    l = choseLeader(memList);//选择新组长
    notifyNewLeader(memList);
    notifyMembders(l);
return;
```

算法 4:主持竞选算法

2.9 处理稳定网络规模的变化

当覆盖网规模发生重大变化的时候需要相应改变所使用 CCC 的维度,下面是改变的方法。

假设网络规模变为原来的一半,则仅需要将环标识相同,立方体标识仅最后一位不同的组合并即可。以标识符为 $(k, a_{d-1}a_{d-2}\cdots a_10)$ 的 X 组和标识符为 $(k, a_{d-1}a_{d-2}\cdots a_11)$ 的 Y 组为例。X 组的组长 x 向 Y 组组长 y 发出一条合并请求,这条信息仅一跳就能到达 y 。合并完成后 x 将成为新组 $(k, a_{d-1}a_{d-2}\cdots a_1)$ 的组长。这条信息中包含 x 的 IP 地址、分配给 y 的成员号(小于 X 组当前的最小成员号)、X 组当前的最大成员号(Y 组成员将使用将该值与自己当前成员号相加的方式得到其新组成员号)以及 X 组保存的对象数据。收到该信息后 y 记录 x 的 IP 和自己的新成员号以及 X 组保存的对象数据,成为以 x 为组长的新组的最“老”节点。然后它将该信息中的自己的新成员号删除,并转发该自己原来的成员。最后 y 还要回复给 x 一条包含 Y 组成员列表(对应的成员号已经做出了改变)和保存的对象数据的信息,以供 x 和 X 组成员做出相应的升级。

网络中所有的节点仅需要将路由表中所记录的节点标识符的最后一比特删除而无需对路由表进行更新。路由表条目的更新会被推迟到该条目被使用一次后,而且这显然不会影响应用。当某一条目被用到时,

如果对应的节点已不再是组长节点,那么它会在完成查寻请求的同时通知源节点进行路由更新。

相反,如果覆盖网规模加倍,每一个组将分裂为两个。以标识符为 $(k, a_{d-1}a_{d-2}\cdots a_1)$ 组长为 x 的组为例。它将分裂为标识符为 $(k, a_{d-1}a_{d-2}\cdots a_10)$ 和 $(k, a_{d-1}a_{d-2}\cdots a_11)$ 的两个组。 x 为前者的组长,而原组中最老的节点成为后者的组长。

3 仿真实验及分析

这一部分将通过与 Cycloid 的对比仿真实验和理论分析,对 CT-Cycloid 的性能进行评价。

(1)路由和定位的开销。CT-Cycloid 的网络直径为 $O(\log S)$ 的,所以其路由和定位的开销是 $O(\log S)$,比一般的结构化 P2P 中的 $O(\log N)$ 的开销要小(这里 $S = N/\log N$)。

在网络规模同为 $\log N \times N$ 的情况下,设 Cycloid 查询路径长度 $O(d_1)$,CT-Cycloid 的查询路径长度为 $O(d_2)$,会有下面两式成立。

$$\log N \times N = d_1 \times 2^{d_1} \quad (1)$$

$$N = d_2 \times 2^{d_2} \quad (2)$$

令 $\log N$ 的底数为 2,由上面两式可得出 d_1 和 d_2 的关系为: $d_1 = \log_2 d_2 \times d_2$ 。也就是说 CT-Cycloid 与 Cycloid 相比查询路径更短。

仿真实验的结果如图 2,在不同的网络规模之下 CT-Cycloid 所需的查询路径长度总是比 Cycloid 小。

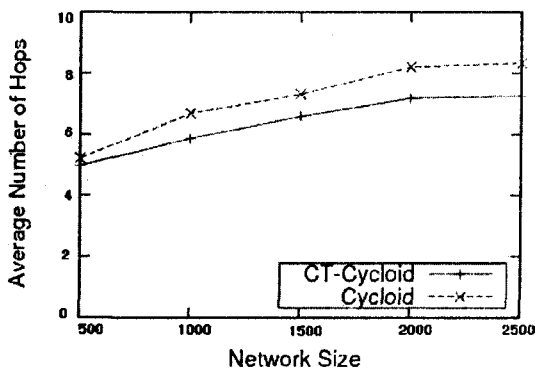


图 2 CT-Cycloid 和 Cycloid 在不同网络规模下的查询路径长度

(2)抗 churn 能力。这一部分我们做了一些仿真实验验证 CT-Cycloid 的抗网络 churn 能力。

试验设置借鉴参考文献[12]。开始时每个节点活跃的概率为 0.5,每个活跃节点使用指数分布来计算自己的存活时间,该指数分布有一个被称作“平均活跃时间”的参数。该参数的值被设置的越小,产生的 churn 就越大。节点执行一次查询结束到下一次查询开始的时间间隔也被使用指数分布模拟,其平均值用

tbs 来描述。实验中“平均活跃时间”值分别被设置为 60s,300s,600s,3000s 和 6000s,tbs 被设置为 5s。图 3 是两种系统在不同的 churn 率下查询命中率情况。图 4 是两种系统在 churn 存在的情况下查询成功所需要的平均路径长度。图 5 是两种系统在不同 churn 率下查询成功时花费的平均时间分布。

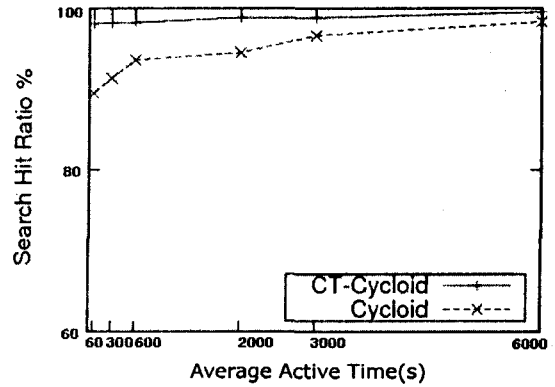


图 3 不同 churn 环境下文件查询成功率

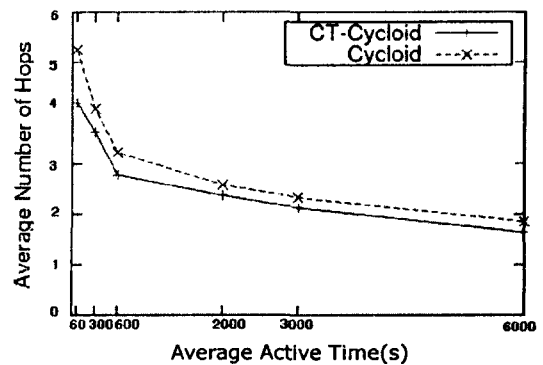


图 4 不同 churn 环境下文件查询成功所需的平均路径长度

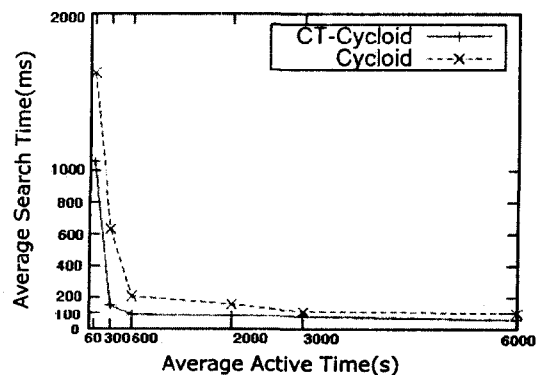


图 5 不同 churn 环境下文件查询成功所需的平均时间

这三个图显示,在网络存在 churn 特别是高 churn 的情况下 CT-Cycloid 比 Cycloid 具有更高的文件查询命中率,查询成功时需要更少的跳数和时。这些都证明 CT-Cycloid 比 Cycloid 具有更强的抗网络 churn 能力。

4 结束语

介绍了一个新的具有高抗 churn 能力的常数度 P2P 系统——CT-Cycloid。通过理论分析和仿真实验证明了 CT-Cycloid 具有以下优良特性:路由和定位开销低 ($O(\log S)$), 一般结构化 P2P 系统为 $O(\log N)$, 其中 $S = N/\log N$ 、查询跳数少, 与 Cycloid 相比在 churn 环境下查询具有更高查询命中率、更少的平均路径长度和更少的查询时间。

下一步工作是通过仿真实验检验 CT-Cycloid 的负载均衡情况以及在稳定的网络规模发生较大变化的时候, CT-Cycloid 系统能否及时有效地演变来保证网络的正常运行。

参考文献:

- [1] 陈贵海, 李振华. 对等网络: 结构, 应用和设计[M]. 北京: 清华大学出版社, 2007.
- [2] 张联峰, 刘乃安, 钱秀祺, 等. 综述: 对等网(P2P)技术[J]. 计算机工程与应用, 2003, 39(12): 142-145.
- [3] 吴国庆. 对等网络技术研究[J]. 计算机技术与发展, 2008, 18(7): 100-103.
- [4] 杨冬, 董平, 张宏科. 对等网络 Churn 问题评估模型与分析[J]. 通信学报, 2007, 28(6): 39-47.
- [5] 张宇翔, 杨冬, 张宏科. P2P 网络中 Churn 问题研究[J]. 软件学报, 2009, 20(5): 1362-1376.
- [6] Shen H Y, Xu C Z, Chen G H. Cycloid: A new constant degree and lookup efficient P2P overlay network[C]//In: Proceedings of International Parallel and Distributed Symposium (IPDPS'04, Santa Fe). [s.l.]: [s.n.], 2004.
- [7] 陈贵海, 须成忠, 沈海英, 等. 一种常数度数的覆盖网络结构[J]. 计算机学报, 2005, 28(7): 1084-1095.
- [8] Stutzbach D, Rejaie R. Understanding churn in peer-to-peer networks[C]//In: Proc. of the 6th ACM SIGCOMM on IMC. New York: ACM Press, 2006: 189-202.
- [9] Bustamante F E, Qiao Y. Friendships that last: Peer lifespan and its role in P2P protocols[C]//In: Proc. of the 8th Int'l Workshop on Web Content Caching and Distribution (WCW 2003). Norwell: Kluwer Academic, 2003: 233-246.
- [10] Bustamante F E, Qiao Y. Designing Less-structured P2P Systems for the Expected High Churn[J]. In IEEE/ACM Trans. on Networking, 2008, 16(3): 617-627.
- [11] Pandurangan G, Jagannathan S. A Simple Churn-Tolerant Structured Peer-to-Peer Scheme[EB/OL]. 2008. <http://www.cs.purdue.edu/homes/gopal/p2p-final.pdf>.
- [12] Roderio-Merino L, Ant A F, Lpe L, et al. Self-managed topologies in P2P networks[M]//In: Computer Networks: The International Journal of Computer and Telecommunications Networking. North-Holland, Inc: Elsevier, 2009.

(上接第 243 页)

市场上其他视频监控系统相比, 开发周期短、价格低廉, 适用于对视频图像要求不高的场合。

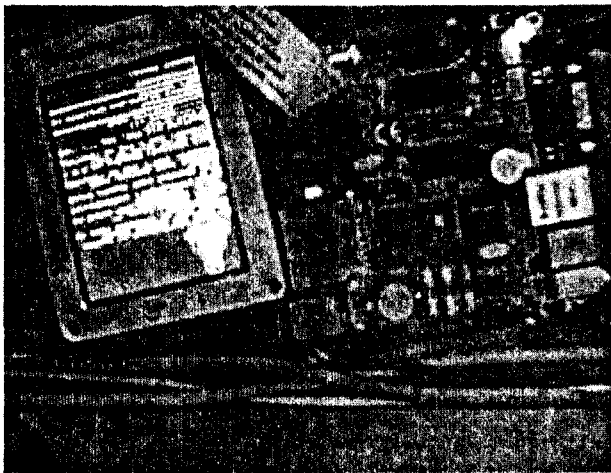


图6 Server端实时视频图像

参考文献:

- [1] 李侃, 廖启征. 基于 S3C2410 平台与嵌入式 Linux 的图像采集应用[J]. 微计算机信息, 2006, 22(2-3): 125-127.
- [2] 嫣红国. 三层客户/服务器结构信息系统优化技术探讨[J]. 微机发展(现更名: 计算机技术与发展), 2002, 12(4): 27-29.
- [3] Samsung Electronics. S3C2440A user's manual preliminary[M]. revision 0.14. [s.l.]: [s.n.], 2004.
- [4] 韦东山. 嵌入式 Linux 应用开发完全手册[M]. 北京: 人民邮电出版社, 2008.
- [5] ATMEL Co. Ltd. Low-cost USB Hub Controller AT43301[M]. [s.l.]: [s.n.], 1998.
- [6] 冯世奎. 基于 ARM 的 Linux 嵌入式系统移植的研究与应用[D]. 成都: 电子科技大学, 2006: 1-2.
- [7] 李明. ARM Linux 的移植过程及分析[J]. 电子设计应用, 2003(7): 55-57.
- [8] 刘振纲, 刘成安, 卢剑翔. 移植标准 Linux 到 S3C2410[J]. 微计算机信息, 2006(11): 152-153.
- [9] Yaghmour K. 构建嵌入式 LINUX 系统[M]. 北京: 中国电力出版社, 2004.
- [10] 胡永红. 智能多路视频监控系统的的设计[J]. 微机发展(现更名: 计算机技术与发展), 2001, 11(2): 75-76.
- [11] Yang L T. Embedded software and systems[M]. Berlin: Springer, 2005.
- [12] WALL K. GNU/Linux 编程指南[M]. 北京: 清华大学出版社, 2002.
- [13] 倪继利. Linux 内核分析及编程[M]. 北京: 电子工业出版社, 2005.