设计文档

一、 资源发现方案设计

在满足题目要求的基础上，为了使不同局域网中用户也能相互发现资源，同时考虑到带宽占用，系统容错能力，编码难度等问题，我们采用的借助服务器（Tracker）实现资源发现的方案。在本方案中，Tracker 服务器不用保存实际的资源信息，可以减少对服务器的性能的依赖，同时也可以避免法律等问题。

1. 总体方案

在本设计中，每个用户（Peer）维护一个本地所有资源信息的列表（本地资源信息列表），当其他用户查询所有在线资源信息时，用户向他们发送这些信息。不同的用户可能拥有相同的资源，由于本题并不考虑实际资源内容的传输，所以在测试的时候可以先假设某些节点拥有相同的资源而不用关心这些资源的是如何获得的。对于没有任何资源的用户，其本地资源列表为空。

拥有资源的用户（资源用户）向 Tracker 发送在线消息，Tracker 维护一个在线资源主机的列表（在线资源主机列表）。如果一个用户需要获取网络中所有的可用资源信息，它先向 Tracker 请求一份最新的在线资源主机列表，再根据列表提供的信息，向每一个资源用户请求资源信息。从而获得网络中所有资源的信息。如图 1 示

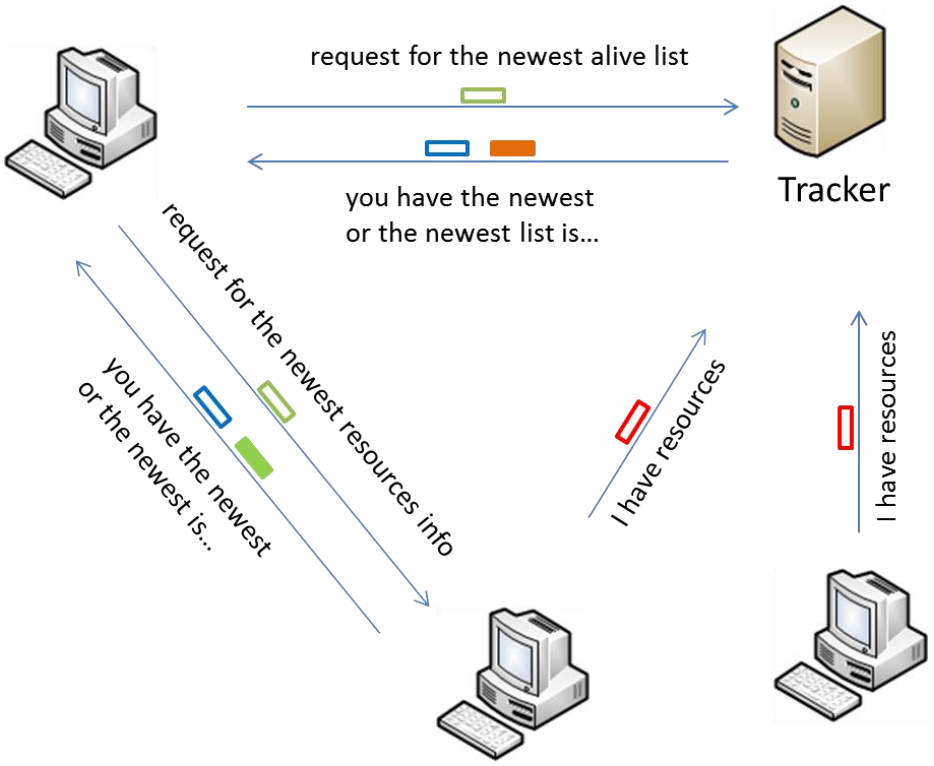

图 1

2. 传输优化

如果用户频繁地获取网络中所有资源的信息，Tracker 和资源的服务的负担会很大。考虑到现实环境中，一个用户的资源不会在短时间内发生剧烈的变化，拥有资源的在线主机也不会剧烈的变化，我们采用版本号的方式来标识在线资源主机列表和本地资源列表。

Tracker 维护一个标识在线资源主机列表的版本号，当用户向 Tracker 请求数据时，用户先发送自己所拥有的列表的版本号，Tracker 把接收到的版本号和最新的版本号比较，如果相同，说明用户持有的在线资源主机列表已经是最新的，不需要改变，所以回传输一个特殊的空消息给用户，如果不同，说明列表已经更新，则发送最新的列表信息和版本编号给用户。这样可以有效的减少不必要的数据传输，降低带宽的占用。

同样，每一用户也维护一个标识自己所拥有的资源信息列表的版本号，采用相同的机制减少不必要的数据传输，降低网络带宽的占用，提高响应的速度。

3. 实现思路

实现中，拥有资源的 Peer 使用 UDP 报文向 Tracker 服务器发送在线消息，Peer 和 Tracker 之间在线资源主机列表，Peer 和 Peer 之间信息列表的请求和传输采用即时的 TCP 连接，当需要传输时建立 TCP 连接，当传输结束后，自动断开 TCP 连接，降低资源的占用。

Peer：

每一个用户维护一个本地资源的信息列表（没有资源的用户该列表空），并为列表设定一个版本编号（实现上采用的是一个 32bit 的无符号整数），用户同时开启一个侦听端口，监听其他用户的资源列表请求。

当一个用户收到其他用户的请求时，先判断上次相应资源信息列表请求到当前时刻这段时间内资源信息列表是否有改变，如果改变，版本号增加 1，否则保持不变，然后再比较请求用户所发送的版本号是否和最新的版本号相同，如果相同，发送空消息给请求用户，否则把新版本号和资源信息列表发送给用户。

拥有资源的主机定时的向 Tracker 发送在线消息，消息的内容为资源主机的请求监听端口（实现上是一个 16bit 的无符号整数），Tracker 把 IP 地址和监听端口号记录到列表中。

当 Peer 需要获取网络中所有在线资源的信息时，Peer 向 Tracker 发送请求消息，消息的内容为自己拥有的在线主机列表版本号（32bit 无符号数），Tracker 根据版本号判断时候需要传最新的列表和版本号给 Peer 用户。

Peer 依次向拥有资源的在线主机请求其所拥有的资源信息。

Tracker：

Tracker 维护一个在线资源主机和其标记的列表，Tracker 接收到 Peer 的在线消息后，把 Peer 的 IP 地址和监听端口（消息的内容）构成一个关键字，如果在线主机列表中没有该关键字，则添加并设置其的标记为真；如果存在则更新其标记为真。Tracker 定时的检查列表中的项，如果某项关键字的标记为假，证明在一个检查周期内没有收到其在线消息，推测其

不在线，故删掉。对于标记为真的项，设置其标记为假，为下一次的检查提供初始值。

Tracker 收到 Peer 的在线资源主机请求消息时，先判断上次响应在线主机列表请求到当前时刻这段时间内，在线资源主机列表是否有过变化，如果有则版本号增加 1。Tracker 比较 Peer 在线资源主机列表的版本号是否和最新的相同，如果相同则发送空消息给用户，否则发送最新的在线资源主机列表和版本号给 Peer。

在对版本号的设定上，Peer 和 Tracker 都采用懒惰的改变改变策略，即需要改变时将其改变，这样可以有效的降低版本号改变的频率，增加版本号出现重复的时间，降低因为版本号有位数有限，出现的重复问题而带来问题的可能。

相关主要策略如图 2-6 示：

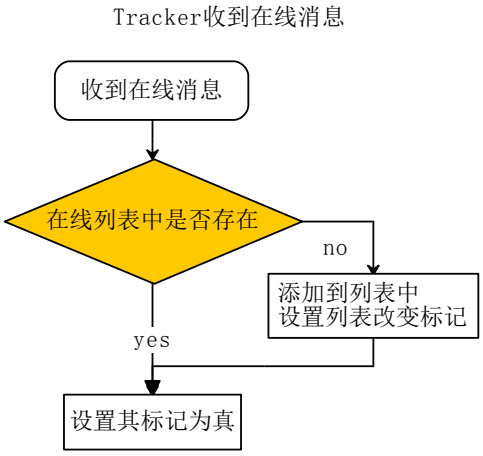

图 2

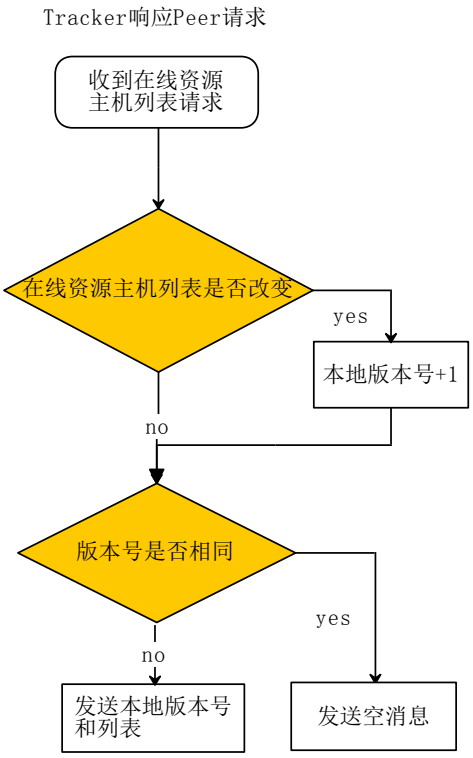

图 3

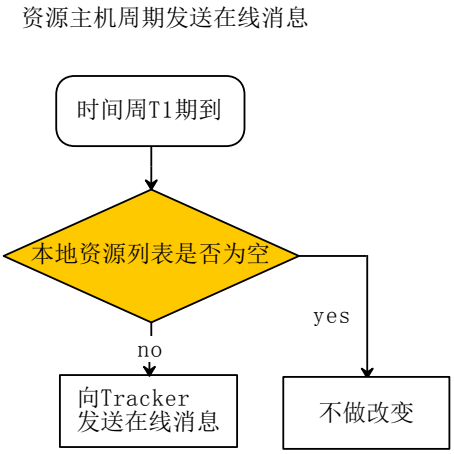

图 4

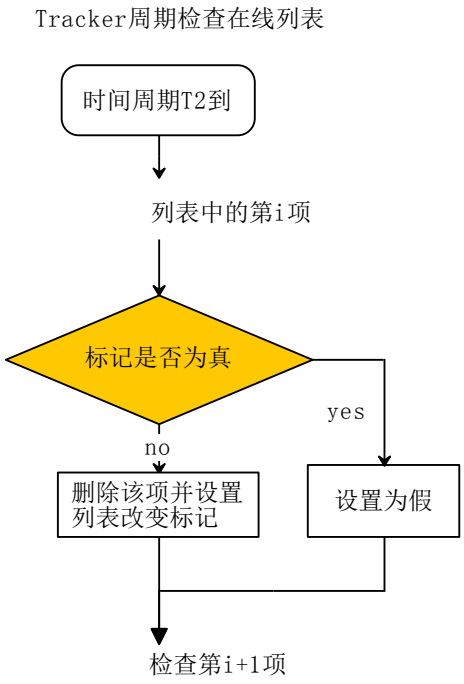

图 5

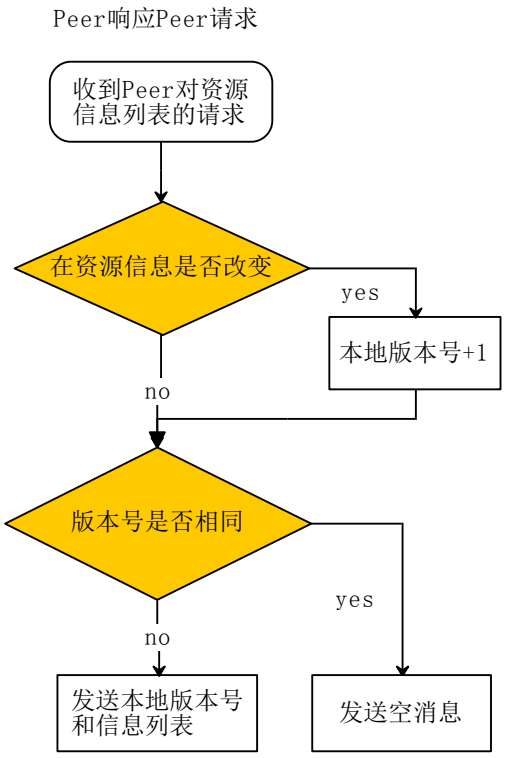

图 6

二、 系统框架

Tracker 主要有这几部分功能模块构成，如图 7 示。

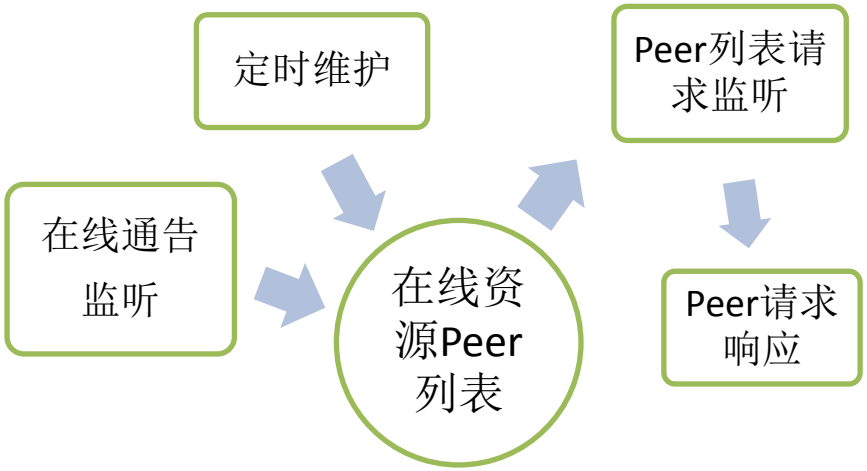

图 7

在线通告监听负责拥有资源的 Peer 定时发送的在线消息，更新在线资源主机列表中 Peer 的状态标志。定时维护模块周期的检测在线资源主机列表中用户的标志，根据在两次维护之间是否收到在线通告来决定资源主机的状态，有此来判断主机是否下线。定时维护的

周期大于资源主机在线消息的发送周期，合理的设置两个周期的时间可以有效的减少拥有资源的主机发送报文的数量。由于本题对实时性有要求，我们设定 Peer 广播的周期为 2s，Tracker 定时维护的周期为 4s。

Peer 主要构成如图 8-9 示

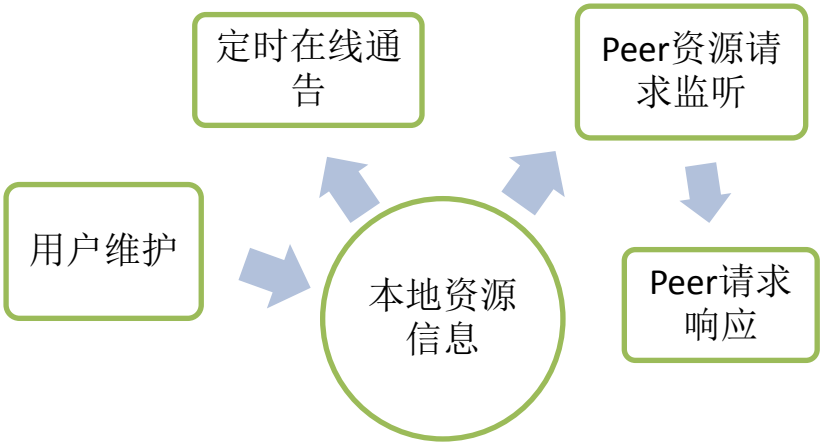

图 8

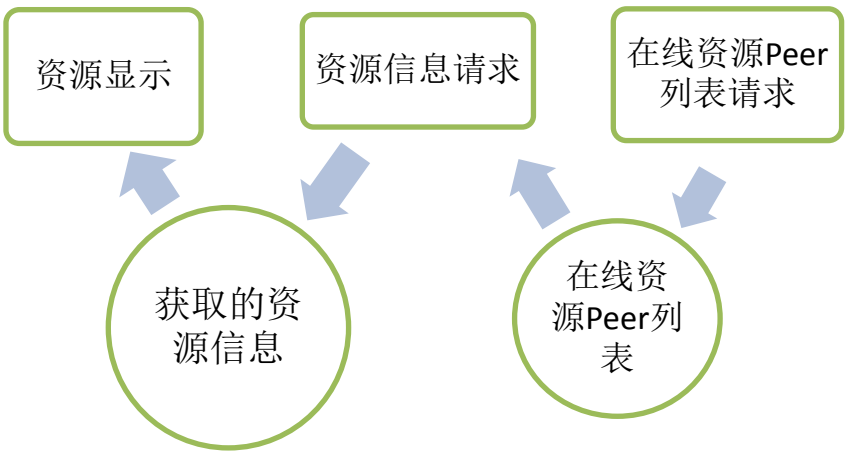

图 9

图 8 所示部分负责管理本地资源信息并共享响应其他 Peer 的资源信息请求，承担起服务器的功能，图 9 负责获取从网络中所有在线资源的信息并显示。

三、 代码实现

软件采用 QT4 编程实现。

1. 资源的表示

我们采用 128bit 的 ID 值唯一的标识一个资源，同时还保留资源的大小（size），名字（name），类型（type）等属性，同时使用 other 字段附带额外信息。资源的 ID 值根据资源

的属性和内容进行 MD5 计算获得。软件采用对 size，name，type 和 other 值进行 md5 计算获得 ID 值，由于 MD5 的散列特性，不同的资源 MD5 值发生碰撞的可能很小，基本可以保证 ID 值在网络中的唯一性。

由于在在程序中采用了版本号的方案减少资源的传输，所以每一个 Peer 需要维护一个资源拥有者到资源信息的映射，由于不同的用户可以拥有相同的资源，为了显示资源信息和持有者信息，并维持好的响应速度，程序同时还维护了一个资源到 peer 的映射，在程序中，我们采用 multimap 容器实现。资源的名字，大小，类型和 other 字段又可能会是很长的数据，如果映射每个对应资源信息的地方都保存完整的资源信息会占用大量的内存，我们为资源信息再建立一个 ID 到属性的映射，同时记录资源被引用的次数（引用计数器），当需用资源信息做键值时，直接用其 ID 值代替，并对其应用计数器加一，当删除某键值时，对于的引用计数器减 1，如果一个 ID 对应的引用计数器为 0，表示该资源没有被引用，可以从映射中删除。程序中有关资源信息相关的数据结构保存在头文件 resource.h 中，相关类的实现和函数重载在 resource.cpp 中。在程序中，还使用了 List，Pair，Map 等容器。

在 Qt 中，我们采用 QByteArray（数组）存储 ID（大小为 128bit），quint32（32 位无符号整数）存储 size，用 QString 存储 name 和 type，同时用 QByteArray 存储 other 字段。

2. 数据同步

我们把软件的各个模块写成了类或者函数，在运行的过程中，有的模块之间会进行通信或者数据共享，我们采用了内存共享的方式来实现。

第一种实现方案是在程序启动时开辟一段公共的内存空间，提供给不同的模块访问和操作。比如，模块 A 负责对数据 D1 进行维护，模块 B 负责处理其他的用户对 D1 的请求。程序在启动的时候申请一段全局的内存标识，让 A 和 B 都能访问这段内存。Tracker 对在线资源主机列表的维护和响应 Peer 的请求， Peer 对资源信息列表和维护和响应用户的请求就是使用的这种方式。当然内存的标识也可以通过信号和槽机制的参数传递。

第一种方案是根据需要动态的申请内存空间存储数据，利用 QT 信号和槽的机制吧内存标识作为信号和槽的参数传递，从而实现消息的传递。在这种方案中，动态申请的内存在不需要要使用后，需要显示的释放，负责会带来内存泄漏的问题。实现消息的传递和数据的共享，比如，在 Peer 中，我门把添加新资源的操作写成了单独的模块，当需要添加新的资源信息时，我申请一段内存存储资源信息的内容，并把内存的标识用信号传给维护资源信息列表的模块，维护模块负责把新的资源信息添加到列表中并释放该段内存空间。

3. 主要类的介绍

程序的各功能模块被单独写成了类或者函数，程序在运行的过程中根据需要动态的创建对象实体实现相应的功能。他们被定义在不同的头文件中。

主要的类和相关的头文件介绍如表 1-4 所示。

编号	名称	主要功能	父类	备注
1	AliveAnnounceReciver	接受通告，更新在线主机列表	QUdpSocket	是 5 的成员
2	PeerRequestHandler	响应 Peer 的在线主机列表请求	QTcpSocket	3 动态创建
3	AlivePeerListServer	监听 Peer 的在线主机列表请求	QTcpServer	是 5 的成员
4	MainWindow	图形界面相关	QWidget	是 5 的成员
5	Tracker	设置共享数据，创建启动 1，3，4	QObject	

表 1：Tracker 主要类介绍

编号	名称	内容	定义的类
1	window.h	Tracker 界面相关的定义	4
2	tracker.h	Tracker 核心功能相关类的定义	1，2，3，5

表 2：Tracker 头文件作说明：

编号	名称	主要功能	父类	备注
1	resource_propertie	表示一个资源信息的属性值		是 2 的成员
2	Resource	表示一个完整的资源信息项		
3	AliveAnnounce	有资源时，向 Peer 发送在线消息	QUdpSocket	6 动态创建
4	AliveListRequester	从 Tracker 获得最新在线主机列表	QTcpSocket	5 动态创建
5	ResourceRequester	向 Peer 请求其资源信息列表	QTcpSocket	6 动态创建
6	AliveResourceInfoUpdater	获取网络中的所有资源信息	QObject	10 动态创建
7	PeerResourceSender	响应 Peer 的资源信息列表请求	QTcpSocket	8 动态创建
8	PeerServerDemo	监听 Peer 的资源信息列表请求	QTcpServer	10 动态创建
9	PeerWindow	界面相关	QWidget	10 创建
10	Peer	设置共享数据，创建并启动模块	QObject	

表 3：Peer 主要类介绍

编号	名称	内容	定义的类
1	include_file.h	常用到的 Qt 头文件	
2	resource.h	资源信息表示相关类的定义	1，2
3	peer_resource_server.h	资源信息发布相关类的定义	7，8
4	alive_announce.h	在线消息通告相关类的定义	3
5	alive_resource_list_updater.h	资源信息获取相关类的定义	4，5，6
6	tabPageLocalSource.h/tabPageAllSource.h peerWindow.h/peersList.h/ deleteSourceWindow.h/addSourceWindow.h	图形界面相关（包含了多个类的定义）	9
7	peer.h	Peer 各功能模块的定义	10

表 4：Peer 头文件说明