基于 MQTT 协议的消息推送服务器①

任 亨[1,2], 马 跃[1], 杨海波[1], 贾正锋[1]

[1](中国科学院 沈阳计算技术研究所, 沈阳 110168)
[2](中国科学院大学, 北京 100049)

摘 要: 以 ipad、iphone 以及 android 手机为代表的移动设备和智能终端近年来取得了巨大的发展, 其用户数迎来了爆发式的增长. 为了适应实际应用中对信息获取及时性, 耗电量, 以及网络环境等方面的需求, 需要用推送的方式取代传统拉取的方式来进行消息的传递, MQTT(Message Queuing Telemetry Transport, 消息队列遥测传输)就是专为这种情况所设计的一种消息传递协议. 本文介绍了 MQTT 协议的基本内容和特点, 以 Mosquitto、Redis 等开源项目为基础设计并实现了一个基于 MQTT 协议的消息推送服务器, 能够对用户订阅的消息进行推送, 同时还实现了用户身份验证、ACL 权限检查、自动订阅话题、热点话题统计、服务器状态监控等功能.

关键词: 移动社交网络; 消息推送; MQTT; Mosquitto; Redis

Message Pushing Server Based on the MQTT Protocal

REN Heng[1,2], MA Yue[1], YANG Hai-Bo[1], JIA Zheng-Feng[1]

[1](Shenyang institute of computing technology, Chinese academy of sciences, Shenyang 110168, China)
[2](University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Mobile devices and smart terminal represented by ipad, iphone and the android phones have made a significant development in recent years. The number of their users is growing explosively. In order to meet the requirments of timeliness of accesses of information, power consumption, as well as the network environment in practical application, we should replace the traditional pull way with the push way to deliver messages. MQTT(Message Queuing Telemetry Transport) is a messaging protocol just designed for the situation. This paper describes the basic content and features of the MQTT protocol.Meanwhile it designs and realizes a message pushing server based on the MQTT protocol in foundation of several open source projects such as Mosquitto and Redis.The server can push messages to the users who have subscribed them before.In the meantime, some more functions such as user identity authentication, ACL permission checking, topic automatic subscribing, hot topic stastics and server status monitoring are added.

Key words: mobile SNS; message pushing; MQTT; mosquitto; redis

随着移动互联网和智能终端的普及, 移动社交网络, 如微信、新浪微博、人人网、Facebook 等迎来了爆炸式的发展, 这对移动终端设备上信息的获取提出了较高要求, 主要体现在移动性和时限性两个方面. 移动性, 要求低功耗、低速率条件下的消息传输; 时限性, 要求信息在规定时间内发送到移动终端设备上. 原有的“拉取”的方式, 需要用户或者应用程序不停的检查新信息的更新通知, 然后向服务器发出拉取请求以获取消息[1], 而“推送”的方式则由服务器主动将新更新的信息直接发送给用户, 减少了交互的次数和负担, 缩短了反应时间, 提高了效率, 重要的是它无需用户参与, 使用户在第一时间即时的获知自己所需要的信息和内容[2,3]. 推送技术在移动终端上的重要应用性逐步凸显出来.

① 收稿时间:2013-08-08;收到修改稿时间:2013-09-30

目前在IOS和Android平台上都有自己的推送系统, 但由于在网络、操作系统和应用方面的限制, 在使用上都有一定局限性. Google 的云消息服务受 Android 版本限制(必须大于 2.2 版本), 且该服务在国内不够稳定, Iphone 上的 APNS 也仅适用于 IOS, 无法跨平台推送.

本文基于MQTT协议设计并实现了一个消息推送服务器, 第一部分介绍 MQTT 的相关内容, 第二部分介绍了系统的框架设计, 第三部分详细阐述了系统中各个模块的设计与实现, 第四部分对服务器进行了功能和性能上的测试, 最后对全文作出总结, 并对下一步的工作进行了展望.

1 MQTT协议

1.1 MQTT 协议的背景

MQTT 协议由 IBM 和 Eurotech 公司于 1999 年开发, 是一套轻量级跨平台的基于发布/订阅的消息传输协议. MQTT 是专门为低带宽、不稳定网络以及计算和处理能力受限的设备所设计的, 协议采用小型传输, 耗电量小, 能大大降低网络流量, 最小化数据包并有效分配与传输, 非常适合移动系统上面的应用[4], IBM 公司已经成功将其应用于智能实验室、St.Jude 远程医疗中心等项目, Facebook 在 IOS 上的应用也利用了 MQTT 协议来进行消息传递. 2013 年 3 月, OASIS (Organization for the Advancement of Structured Information Standards, 结构化信息标准促进组织)宣布将 MQTT 作为新兴的物联网消息传递协议的首选标准, 随着物联网时代的来临, MQTT 协议必将获得空前的关注和广泛的应用.

1.2 MQTT 协议的消息格式

MQTT 消息体主要由三部分组成: 固定头, 可变头和有效载荷, 其中只有固定头是所有消息体都必须包含的部分. 其结构如表 1[5]所示.

表 1 MQTT 消息固定头

Bit	7	6	5	4	3	2	1	0
byte 1	Message Type				DUP flag	QoS level		RETAIN
byte 2	Remaining Length							

其中 Remaining Length 表示除固定头之外的消息长度, 最大可扩展到 4 字节, 最大表示长度可达 256MB. MQTT 共包含 14 种消息, 按功能分为连接类、消息订阅/发布类和保活类, 主要的消息类型及其对应数值如表 2 所示.

表 2 部分 MQTT 消息类型

Mnemonic	Value	Mnemonic	Value
CONNECT	1	UNSUBSCRIBE	10
PUBLISH	3	PINGREQ	12
SUBSCRIBE	8	DISCONNECT	14

1.3 MQTT 协议的特点

1.3.1 非常小的通信开销

最短的消息只有两个字节, 将最小化协议本身带来的消息传输代价以降低网络负载.

1.3.2 协议简单, 开放, 易于实现

MQTT 协议采用订阅/发布的消息模式, 提供一到多的消息分发, 降低应用的耦合度; 协议具有良好的跨平台性, 可以在 TCP/IP 以及 Zigbee 网络中应用.

1.3.3 可选的服务质量

根据网络状态和服务要求采取三种不同的消息传输质量等级, 分别是

QoS0: 至多发送一次. 无确认. 可能会有消息的丢失, 用于网络状态比较差并且单次数据的丢失不会影响整体结果的情况, 例如用传感器采集环境数据.

QoS1: 至少发送一次. 一次确认. 能确保发送消息到达对端, 可能会收到重复的消息, 用于网络状态一般的情况.

QoS2: 刚好发送一次. 三次确认, 能确保对端收到且只收到一次该消息, 用于消息的丢失和重复会造成错误结果的情况, 比如计费系统.

1.3.4 遗嘱机制

在客户端连接因为网络状态等非正常原因断开后, 根据用户设置的遗嘱机制, 以发布话题的形式通知可能对该用户状态感兴趣的其它客户端用户.

由于 MQTT 协议具有开放、简单、轻量、易于实现等优点, 因此他特别适用于低带宽, 网络不稳定, 网络代价昂贵以及处理器和存储器资源有限的嵌入式设备和移动终端上.

2 系统框架设计

整个服务器部分主要分成三个层次. 第一层是 MQTT 消息推送 broker, 负责完成协议底层的网络通信机制以及针对各种不同类型消息的收发机制; 第二

层由身份验证模块、ACL 控制模块、自动订阅模块、话题统计模块以及状态监控模块组成, 是在底层通信机制的基础上完善整个系统实际应用中所需的各项功能; 第三层是数据存储层, 为第二层的各个模块提供数据的支持, 用于各项数据的统计与交互. 整个系统框架如图 1 所示.

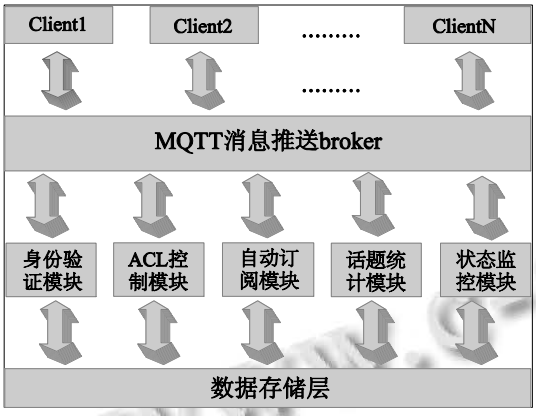

图 1 系统框架

理大并发量下的用户访问请求.

3.3 密码验证模块

在客户端向服务器发起连接请求的时候, 服务器必须对其进行密码验证, 以决定是否接受该连接请求, 验证过程如图 2 所示.

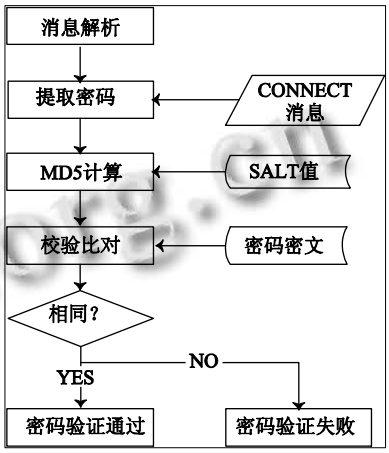

图 2 密码验证流程

3 各模块的设计与实现

3.1 消息推送中间件

目前在各种平台上对于 MQTT 协议有许多不同的实现, 这里所选择的 Mosquitto 是一款开源的基于 C 实现的 MQTT server/broker, 比较完整的实现了 MQTT 协议中要求的各项基本功能, 可以在 Windows, Linux 以及其它类 Unix 系统中编译运行.

3.2 数据存储层

在数据存储层中, 有的数据是需要经常读取的, 比如用户的名称、密码、ID 以及用户间的好友关系等; 有的数据是需要经常写入和修改的, 比如某些话题的订阅数; 有的数据是不经常读取或写入的, 比如服务器的运行状态. 为了提高效率, 对于需要经常读取或写入的与用户相关的数据, 用 Redis 数据库存储, 对于不经常读取或写入的与服务器状态相关的数据, 用 MYSQL 数据库存储.

这里的 Redis 是一个基于 key-value 的开源 no-sql 数据库, 它将数据缓存在内存中, 相对于传统的关系型数据库来说, 性能上有很大提高, 特别适用于对访问速度和并发性要求比较高的情况[6,7]. 同时 Redis 也支持数据的持久化, 并且支持 list, set, hash 等多种不同的数据结构. 在这里采用 Redis 来存储每个用户的用户名密码等相关信息和与话题有关的数量统计, 以处

为了增加数据库的安全性, 在用户注册时需要对其密码加密后再将其密文存入数据库, 这样即使有人利用非法手段侵入数据库, 也无法得到用户的真实密码. 在这里采用的 MD5 加密算法是一种散列加密算法[8], 任何一个密码经过 MD5 的 HASH 散列计算之后将会产生一个 128bit 的序列. 但是缺点是两个相同的密码会产生相同的散列, 为了弥补这个不足, 需要引入 SALT 技术[9], 在用户注册时生成一个随机数据, 与用户密码一起进行散列计算, 然后将得到的密文和 SALT 值一起存入数据库中, 这样就可以保证即使用户的密码相同, 只要随机的 SALT 值不同, 其密文就不相同.

在用户发起连接请求时, 将用户提交的密码明文与数据库中的 SALT 值一起进行散列计算, 将所得密文与数据库中密文进行比对即可对用户密码进行验证.

3.4 ACL 控制模块

为了规范用户行为, 需要对其话题的订阅和发布进行权限控制. ACL(Access Control List)又称为访问控制列表, 是一种通过匹配关系对访问权限进行控制, 以加强系统安全性的技术[10,11].

ACL 表采用的格式为:

user <username>

[read/write] <topic>

或者

pattern [read/write] <topic>

第一种格式为特定用户的权限控制规则, 第二种格式为所有用户的权限控制规则, read 表示具有订阅的权限, write 表示具有发布的权限, topic 采用层级结构组织, 检查的时候从左至右各层依次进行匹配. ACL 表赋予用户应该具有的最小权限, 只要满足表中的一项, 即表示验证通过.

为了满足移动社交网络中的应用需求, 需要引入通配符来表示用户之间的关系, 在这里, 用%u 表示用户自身的用户名, %c 表示用户自身的 ID 号, %f 表示用户的单向关注的关系, %F 表示用户互相关注的关系(这里我们称之为好友), +表示话题中单层的通配, #表示话题中若干层的通配, 一个典型的 ACL 表项为:

pattern read user/%F/presence/+

该规则表示该用户对所有他的好友的 presence 之下的一级子话题具有订阅的权限.

3.5 自动订阅模块

在系统的实际应用中, 用户需要接收各种各样的推送话题, 例如系统的广播通知、好友的上线提醒、好友发送的即时消息等, 如果每次上线的时候都由客户端来对这些话题进行订阅, 不但会影响客户端的性能, 还会占用移动终端的网络资源, 尤其是在用户关系复杂, 需要订阅大量话题的时候. 因此, 服务器可以在用户登录成功之后为用户自动订阅这些话题, 以减少网络上的数据交互. 自动订阅模块流程如图 3 所示.

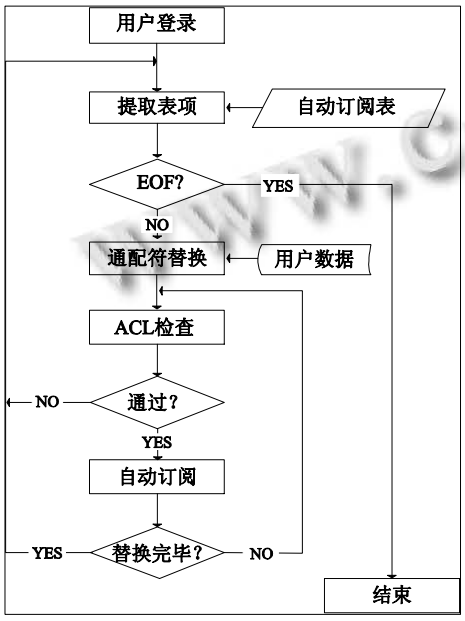

图 3　自动订阅流程

自动订阅支持 ACL 控制模块中提到的各项通配符, 并且将%u 替换为自身用户名, %c 替换为自身 ID, 含有%f 和%F 的表项会被通配为多个话题分别进行自动订阅. 一个典型的自动订阅表项为:

user/%F/presence/+ 1

表示每个用户上线时都会自动订阅其好友的 presence 之下一级的话题, QoS 设为 1, 中间以空格分隔.

3.6 话题统计模块

当大量用户在一段时间内同时订阅某一话题, 或者向某一话题发布消息时, 意味着这个话题是当前用户所关注的热点. 因此, 对话题的统计有助于分析用户群体的行为方式以及当前社会生活中的热点时事, 对于发掘用户的潜在需求有重要的意义.

为了提高消息推动服务器的性能, 话题的统计数据记录在 redis 数据库中, 以 topic-count 的形式存储, 当用户在统计的话题上进行订阅或者发布的时候, 服务器调用 Redis 的 INCR 或 DECR 命令对统计数据进行更新.

3.7 状态监控模块

要确保服务器长时间稳定的运行, 需要定期对系统运行的状态进行监控. 一方面要监控服务器本身的系统信息, 如 CPU 占用率、内存占用率、磁盘剩余空间大小、网络流量等等, 这些数据在 linux 环境下可以通过解析/proc 下相关文件的内容来获取; 另一方面需要监控消息推送服务的运行状况, 如活跃的用户数、用户订阅话题的总数等. Mosquitto 提供了对这些信息进行统计的机制, 所得的数据将由服务器以话题的形式定期进行发布, 监控模块需要以客户的身份登录并订阅相应话题以获取相关信息, 采集到的结果插入到 MYSQL 数据库中, 以供服务器进行性能分析以预警.

4 测试

4.1 功能测试

“沈阳手机广播”是作者所在项目组开发的一款集广播收听、即时聊天、微博、微信等功能于一体的手机 SNS 应用, 同时支持 android 和 ios 平台, 这里把它作为功能测试的终端, 测试流程如图 4 所示.

用户 t1.qq.com 在连接 wifi 的 Android 手机上登录, 用户 yxq_anis.163.com 在连接 3G 网络的 Iphone 上登录,两者为互相关注的好友关系. 自动订阅表项中存在表项.

/%u/chat 1

broker 会为用户 yxq_anis.163.com 自动订阅话题 /yxq_anis.163.com/chat, QoS 为 1. ACL 表中存在表项:

pattern read /%u/chat

pattern write /%F/chat

yxq_anis.163.com 对该话题有订阅权限, t1.qq.com 对该话题有发布的权限, 发布时的QoS为1. 终端上显示结果如图 5 所示.

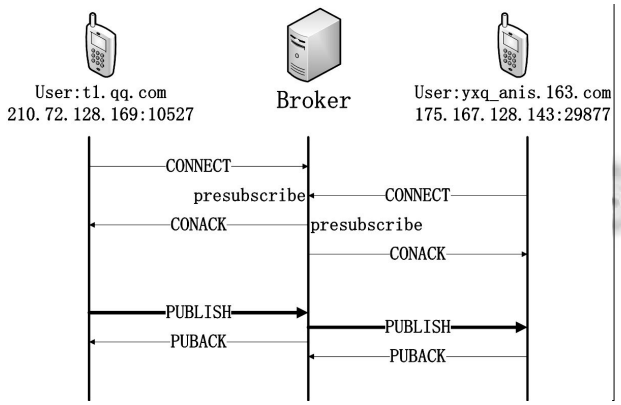

图 4 功能测试流程

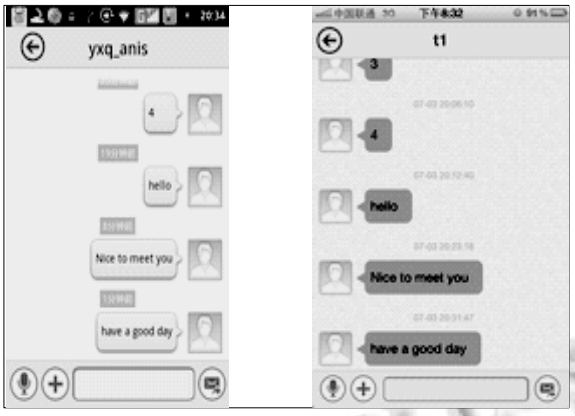

图 5 功能测试终端结果

同时在服务器上利用 tcpdump 命令观察数据包流向, 部分结果如图 6 所示.

其中, 1 至 11 行为 t1.qq.com 向 broker 发布消息的数据包, 15 至 23 行为 broker 向 yxq_anis.163.com 推送消息的数据包, 为保证服务器的安全性, 这里略去了包头的 48 字节并且擦除了服务器的 IP 及端口号.

4.2 性能测试

为了确定服务器在负载逐渐增加的情况下的性能, 需要进行性能测试. 这里在 PC 上实现一个客户端程序来模拟大量的用户对服务器发起连接请求, 服务器

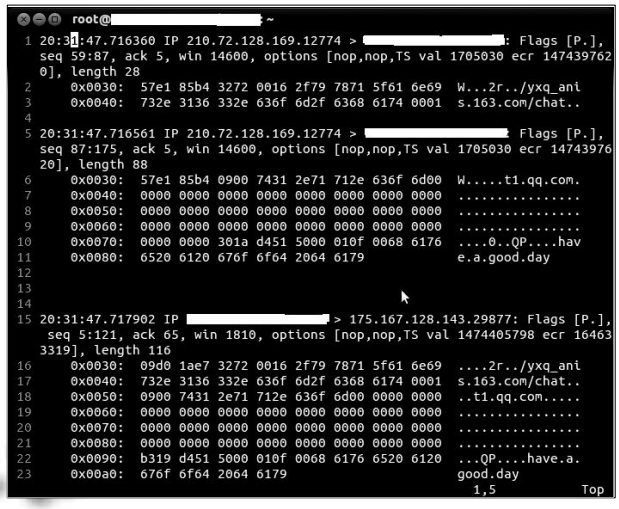

图 6 功能测试部分数据包

端配置如表 3 所示, 网络结构如图 7 所示:

表 3 服务器端配置

CPU	Intel(R) Pentium(R) 4 3.00GHz
内存	2GB
硬盘	450GB
OS	CentOS release 5.5(final)
网卡	100Mbps

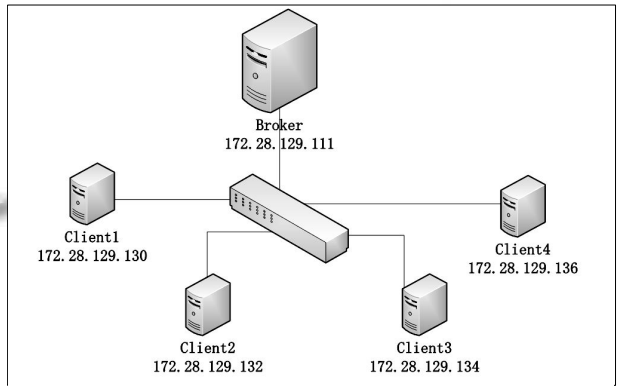

图 7 性能测试网络结构

其中, 每个客户端最多发起 25000 个连接, 在所有连接建立完成后, 分别进行 1 到 100, 1 到 1000,1 到全体的推送, 测试各种情况下所需的时间, 推送成功率及服务器负载情况, 所得结果如表 4 所示.

可以看到, 在局域网环境下, 所有推送的成功率都达到了 100%, 推送的延迟时间主要取决于当前需要推送的消息数, 总连接数只会对其造成轻微影响.

表 4 性能测试结果

连接数	CPU 占用率	内存占用数	1 到 100 推送			1 到 1000 推送			全体推送		
			最快	最慢	成功率	最快	最慢	成功率	最快	最慢	成功率
10000	14.5%	25MB	<1s	<1s	100%	<1s	<1s	100%	10s	14s	100%
20000	25.0%	50MB	<1s	<1s	100%	<1s	<1s	100%	25s	30s	100%
30000	33.2%	76MB	<1s	<1s	100%	<1s	1s	100%	40s	46s	100%
40000	40.1%	101MB	<1s	<1s	100%	<1s	1s	100%	63s	72s	100%
50000	45.1%	126MB	<1s	<1s	100%	1s	1s	100%	104s	118s	100%
75000	55.3%	179MB	<1s	<1s	100%	1s	1s	100%	193s	212s	100%
100000	62.4%	235MB	<1s	<1s	100%	1s	1s	100%	286s	334s	100%

5 总结与展望

本文论述了基于MQTT协议的消息推送服务器的设计与实现, 通过与客户端的联合, 可以对不同平台上的用户进行实时、准确的消息推送, 目前已应用于实际产品中, 经过测试证明, 系统运行良好, 并且可以支持 10 万以内的连接并发数.

下一步的工作将集中在通过建立服务器集群来支持更多的用户上, 同时进一步加强服务器的可靠性及稳定性.

参考文献

1 刘军霞,熊选东,付建丹.基于发布/订阅的推模式服务调用.计算机系统应用,2012(12):196−199.

2 李小智.基于消息中间件的服务器推送技术的应用研究[学位论文].长沙:湖南大学,2010.

3 梅蕊.跨服务器消息发布与推送机制的研究[学位论文].武汉:华中科技大学,2011.

4 IBM. MQ Telemetry Transport.http://mqtt.org.2013-06-05.

5 IBM, Eurotech. MQTT V3.1 Protocol Specification. http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html, 2010-08-24.

6 唐诚.Redis 数据库在微博系统中的实践.厦门城市职业学院学报,2012,14(3):55−59.

7 Schram A, Anderson KM. MySQL to NoSQL: data modeling challenges in supporting scalability. Proc. of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity. ACM. 2012. 191−202.

8 魏晓玲.MD5 加密算法的研究及应用.信息技术,2010,(7):145−147.

9 Gauravaram P. Security Analysis of salt|| password Hashes. Advanced Computer Science Applications and Technologies(ACSAT), 2012 International Conference on. IEEE. 2012. 25−30.

10 杨梅,杨平利,宫殿庆.ACL 技术研究及应用.计算机技术与发展,2011,21(6).

11 唐子蛟,李红蝉.基于 ACL 的网络安全管理的应用研究.四川理工学院学报(自然科学版),2009,22(1):48−51.

(上接第 166 页)

protocol architecture for underwater sensor network. ICSMIM. 2012.

6 Du XJ, Lan SL, Liu F etc. Micro-ANP network protocol architecture and simulation implementation. TELKOMNIKA Indonesian Journal of Electrical Engineering, 2013, 11(4):1757−1768.

7 Kim JP, Lee JW, Jang YS, etc. A CDMA-based MAC protocol in tree-topology for underwater acoustic sensor networks. International Conference on Advanced Information Networking and Applications Workshops. 2009. 1166−1171.

8 Hu LM. Distributed code assignment for CDMA packet radio networks. IEEE/ACM Trans. on Networking, 1993. 668−677.

9 Chen HF, Xie L, etc. An improved CDMA-based MAC protocol for underwater acoustic wireless sensor networks. International Conference on Wireless Communications, Networking and Mobile Computing(WiCom). 2011. 1−4.

10 Yu CW, Wu TK, etc. A distributed code assignment algorithm with high code reusability for CDMA-based ad hoc networks. The First International Conference on Wireless Algorithms, Systems, and Applications. 2006. 329−340.