计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(6):76−81 [doi: 10.15888/j.cnki.csa.006929]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于互联网服务器的海量 ZigBee 节点管理系统①

芦晨博

(沈阳工业大学 信息科学与工程学院, 沈阳 110870)
通讯作者: 芦晨博, E-mail: 1063576810@qq.com

摘　要: 为解决海量 ZigBee 节点的组网与集中管理问题, 文章提出一种基于互联网服务器的多 ZigBee 网络结构管理设计方案. 系统将海量节点拆分为多个 ZigBee 网络结构, 各个网络之间的设计不存在耦合, 可摆脱无线传输的距离限制. 每个独立的 ZigBee 网络中除终端节点与 ZigBee 协调器外, 新增一个可接入互联网的控制器来组成内网结构. 内网中的 ZigBee 节点利用 Z-Stack 协议栈实现自组网功能, 控制器主要用于实现节点与服务器之间的通讯. 文章提出中央服务器的概念作为外网结构的核心, 用于维护中央数据库内所有节点的状态信息, 同时对客户端提供 Web 服务. 客户端对于系统内任意节点的查询或控制操作均由中央服务器代理完成. 经测试表明系统运行可靠, 扩展性强, 具有一定参考价值.

关键词: 互联网; ZigBee; Z-Stack 协议栈; Web 服务

引用格式: 芦晨博.基于互联网服务器的海量 ZigBee 节点管理系统.计算机系统应用,2019,28(6):76–81. http://www.c-s-a.org.cn/1003-3254/6929.html

Realization of Massive ZigBee Node Self-Organizing Network Based on Internet Server

LU Chen-Bo

(Institute of Information Science and Engineering, Shenyang University of Technology, Shenyang 110870, China)

Abstract: In order to solve the problem of networking and centralized management of massive ZigBee nodes, this study proposes a multi-ZigBee network structure management design scheme based on Internet server. The system divides the mass nodes into multiple ZigBee network structures, and there is no coupling between each network, which can get rid of the distance limit of wireless transmission. In addition to the terminal node and the ZigBee coordinator, each independent ZigBee network adds a controller that can access the Internet to form an intranet structure. The ZigBee nodes use Z-Stack protocol to realize self-organizing network function, and the controller is mainly used to realize communication between nodes and servers. The proposed concept of the central server as the core of the external network structure is used to maintain the state information of all nodes in the central database and provide Web services to clients. The client's query or control operations on any node are completed by the central server agent. The test shows that the system is reliable and expandable, and has certain reference value.

Key words: Internet; ZigBee; Z-Stack protocol; Web server

引言

在基于 Z-Stack 协议栈的 ZigBee 网络中通过使用 16 位短地址[1]在本地网络中标识设备和在网络中发送数据, 当一个节点加入网络的时候将由它的父节点

① 收稿时间: 2018-12-10; 修改时间: 2018-12-29; 采用时间: 2019-01-08; csa 在线出版时间: 2019-05-25

给它分配短地址[2]. 这个短地址的长度规定了在一个 ZigBee 网络中最多能搭载 65 535 个节点, 而在实际应用中由于 ZigBee 节点芯片的运算能力有限, 加上载波侦听多路访问 / 冲突避免[3](Carrier Sense Multiple Access with Collision Detection, CSMA/CA) 等原因很难达到上万节点的挂载. 在使用全路由节点网络[4,5], 更强性能的协调器或多信道通讯等手段时虽然可有效提高网络中的节点数量, 仍然无法突破短地址带来的上限, 同时又大大增加了系统复杂度. 在物联网蓬勃发展的今天与万物皆可互联的理念下[6], 搭建能够处理大量节点的网络结构意义重大.

本文通过在 ZigBee 节点网络之上利用互联网服务器搭建外网结构, 集中管理各个内网 ZigBee 节点群, 并提供自组网功能. 根据实际业务需求进行节点网络的组建, 服务端将集中管理所有节点数据并对外提供 Web 服务. 在服务端的架构设计上可采用分布式, 搭建集群, 负载均衡等多种应对高并发[7,8]的设计方式, 使得系统性能与节点挂载能力获得极大提升.

1 整体设计

系统整体由终端节点, 节点协调器, 内网控制器, 中央服务器, 人机接口构成五层网络结构.

终端节点、协调器、内网控制器组成内网结构. 其中 ZigBee 终端节点可挂载各类传感器和控制设备与 ZigBee 协调器之间通过利用 Z-Stack 协议栈完成自组网与无线通讯功能. 每个独立的 ZigBee 网络都配有一台直接与外网相连并具有一定存储功能的内网控制器. 协调器与内网控制器可通过串口相连或直接集成在一起. 控制器接受中央服务器的指令信息来驱动 ZigBee 协调器, 并通过广播的方式向终端节点传递指令消息, 同时将终端的状态信息同步至中央服务器.

中央服务器内维护着各个内网系统控制器的 IP 地址和与之相匹配的身份标识. 整个系统中每一个终端节点拥有一个唯一的全局节点 ID(GlobalID), 服务器通过 GlobalID 和控制器 IP 地址可对指定节点发送控制信息. 中央数据库内存储着所有节点的状态信息并可提供 Web 服务供用户客户端访问. 整体结构如图 1.

2 内网结构设计

2.1 终端节点

任意一个内网结构中, 每个终端节点在启动之前会被分配一个身份标识符 (NodeID). 启动设备后在 Z-Stack 协议栈的帮助下会自动寻找当前网络内的协调器, 若匹配成功则加入该网络. 在完成自组网工作后节点进入正常工作状态. 终端节点除自组网任务外仍需在 Z-Stack 协议栈操作系统抽象层 (Operating System Abstraction Layer, OSAL) [9]中注册和初始化两项基本任务.

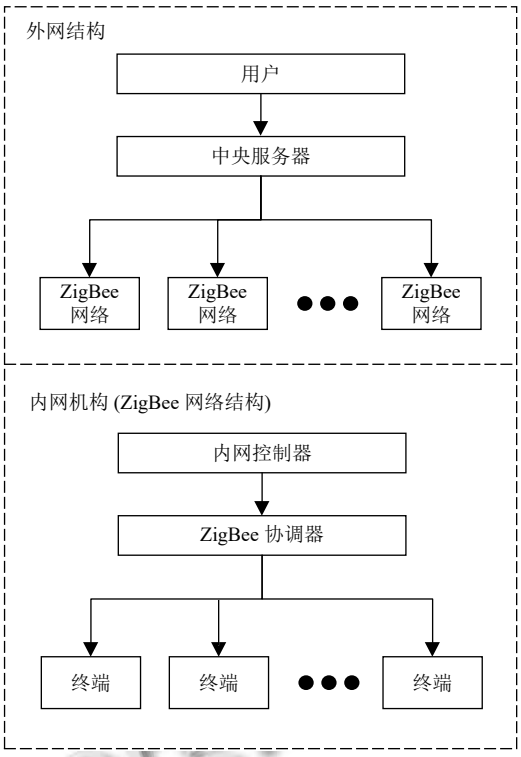

图 1 整体机构示意图

(1) 状态信息定时发送任务: 终端节点周期性的向网络内的协调器发送状态信息其目的是为了表明自身存活状态并可传递各类传感器信息. 数据格式一般为: NodeID+状态信息.

(2) 指令消息接受任务: 系统中协调器以广播的形式传递消息, 数据格式为: NodeID+操作码, 当终端指令与指令消息中的 NodeID 对比成功后执行操作码对应的自定义任务, 对比失败则忽略消息.

2.2 协调器

系统中 ZigBee 协调器除完成自组网任务外, 仅作为消息的传递者. 将内网控制器传来并将受到的终端节点状态信息传递给内网控制器.

2.3 内网控制器

内网控制器是连接 ZigBee 网络与中央服务器的

核心枢纽, 用于解析服务端请求, 向协调器发送指令信息和封装节点状态信息同步至中央服务器. 控制器会被分配一个公网 IP 地址供中央服务器查找. 在本文的设计中控制器需至少包含四项基本任务:

(1) 中央服务器消息监听任务 (CenterMsgListener): 启动一个 SocketService 监听指定端口, 中央服务器会向该端口发送各类指令信息, 调用 MsgParser 解析指令消息, 其中数据段的格式为 GlobalID(全局节点 ID) 的指令消息原封不动通过广播方式发送出去.

(2) 中央服务器消息发送任务 (CenterMsgSender): 新建 SocketClient 向中央服务器发送包括控制器心跳、注册信息、节点状态的更新、新节点入网与节点失活等多种消息.

(3) 协调器消息监听任务 (ZNodeMsgListener): 接收协调器收集到的终点节点状态信息.

(4) 协调器消息发送任务 (ZNodeSender): 向协调器发送正确格式的指令字符. 在控制器初始化阶段还需加载 4 张基本数据表.

表 1 控制器初始化基本数据表

名称	key	value	功能
Global2Node	GlobalID	NodeID	使用全局节点 ID 查找内网节点 ID, 内网节点配置与外网松耦合, 在同一业务背景下可使用相同内网节点 ID.
Node2Global	NodeID	GlobalID	Global2Node 的反向索引表.
NodeStatus	NodeID	Status	存储当前已连接的节点状态信息, 包括控制状态, 传感器状态和上次心跳时间.
Command	GlobalCommand	NodeCommand	中央服务器的指令码与内网节点指令码对照表.

2.3.1 处理服务器控制信息

当控制器接收到中央服务器的指令消息时需依次执行以下任务:

(1) 解析消息获取指令数据段, 解析失败向服务器返回 ERROR-指令消息非法.

(2) 查询 Global2Node 获取内网节点 ID, 获取失败时向服务端返回 EXCEPTION-节点不存在或已下线.

(3) 对照 Command 表封装正确格式的指令信息.

(4) 获取 ZNodeSender 向协调器发送指令消息字符, 若发送失败向服务端返 ERROR (节点操作失败).

(5) 向服务端返回 SUCCESS.

(6) 更新 NodeStatus 表中的状态信息.

(7) 在本地磁盘内新增操作记录与日志信息.

2.3.2 处理协调器节点消息

控制器接收到协调器节点心跳信息时需根据内容作出不同的处理, 当某一节点的心跳消息第一次出现时需在本地缓存节点状态信息, 为减轻服务器端的压力只有在节点状态发生实质上的变化时才会发送更新消息. 具体处理流程如图 2 所示.

2.3.3 处理节点离线

Status 中记录着对应节点的上次心跳时间, 控制器初始化完成后启动监听线程周期性的遍历 NodeStatus 表, 根据当前时间戳查找已超时节点, 封装超时节点信息向服务端发送节点下线消息, 清除 NodeStatus 中缓存的对应节点记录, 更新本地数据库.

3 外网结构设计

3.1 中央服务器

中央服务器是外网结构的核心, 其核心功能主要分为三部分:

(1) 维护各个内网控制器的通讯地址与通讯方式, 快速匹配目标节点发送控制信息.

(2) 响应并处理控制器请求, 存储所有节点状态信息, 维护系统内数据的一致性.

(3) 对外提供 Web 服务, 将对 ZigBee 节点的操作抽象成具体业务功能. 外网结构如图 3 所示.

对于客户端来说, 所有被查询的节点数据均来自于中央数据库, 在享受这种高效的, 体验良好的查询服务时, 需要重点关注数据实时性的问题. 系统内影响实时性的主要因素来源于从节点状态发生变化到数据库更新所产生的时间差.

针对此问题在内网结构中可以选择提高心跳信息的采集频率、使用更高性能的网络模块、在终端节点上采用中断的方式提交更新数据或者可以在同一业务区域内搭建多个内网结构. 更高的实时性必然需要付出更多的成本和性能. 对于中央服务器而言并不需要过多的关心业务, 最重要的是尽可能的提高处理并发的能力. 除了在请求处理上采用合理的负载均衡策略外, 值得一提的是在节点 ID 字段的分配时可在其中加入业务地区与业务类型的标识字符, 这种设计在存储时十分有利于做分库分表[10], 在查询时也可根据 ID 中的标识字符定位数据库的位置. 性能提高十分明显.

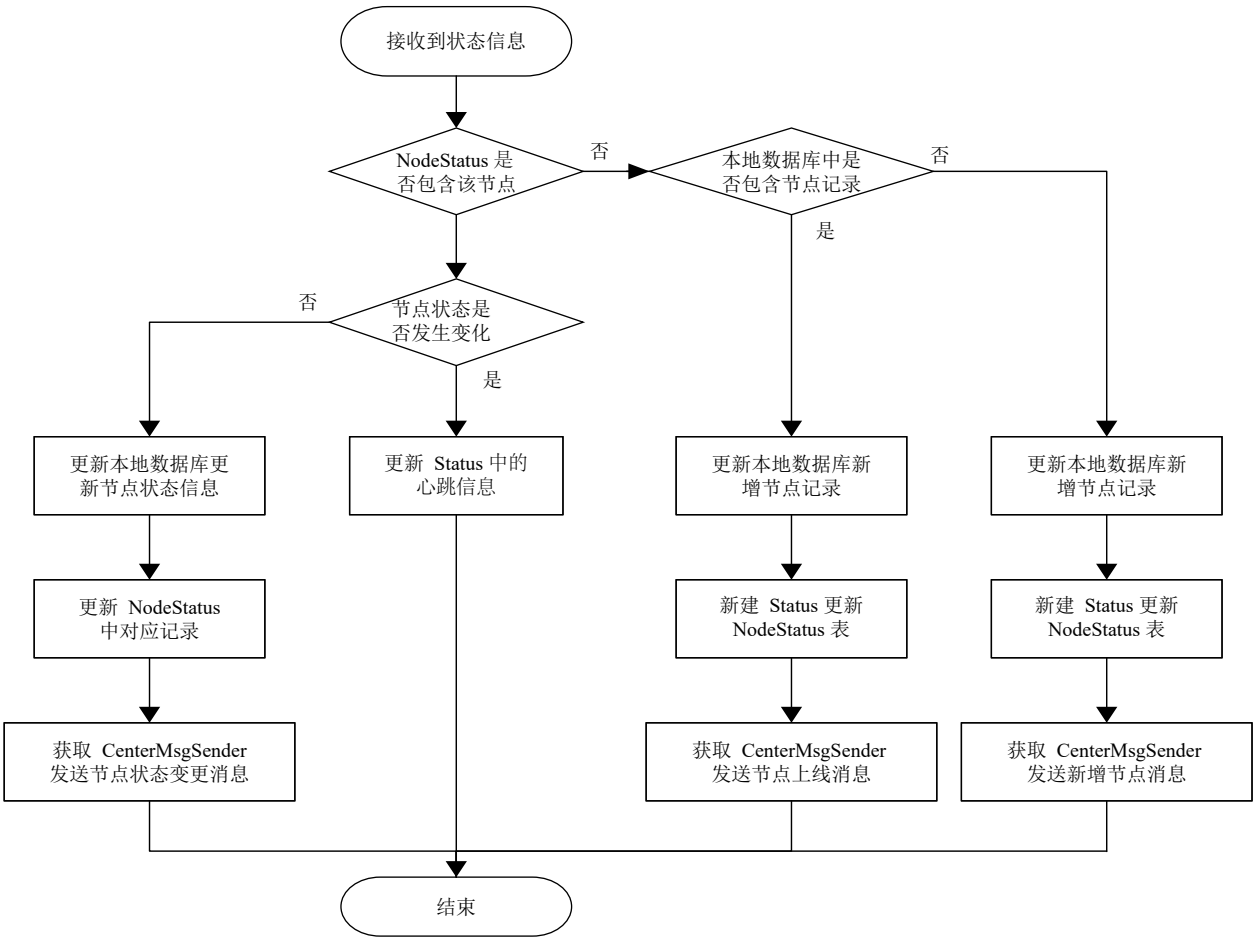

图 2 节点信息处理流程图

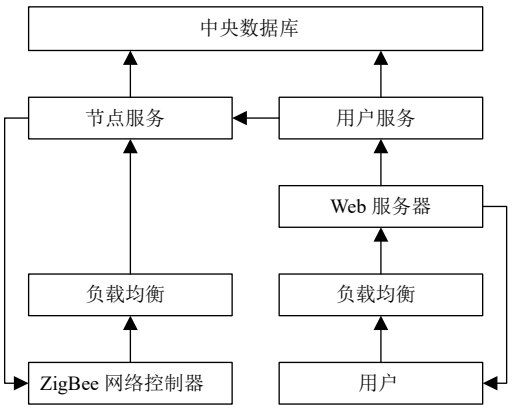

图 3 外网结构示意图

当服务器接受到节点操作请求时, 为保证节点操作的准确性与系统内数据的一致性, 用户的请求与中央服务器对控制器的操作请求都应该是阻塞的, 核心数据的事务[11]应包含整个过程.

3.2 控制器注册与组网机制

在控制器初始化阶段控制器需向中央服务器发送注册信息表明身份请求接入网络. 数据段中至少包括: 自身 IP 地址, 用于接受消息的端口号与通讯的使用密钥.

其中密钥为中央服务器与控制器之间约定的加密与解密字符, 为保证通讯的安全每一次消息的传递都应使用密钥, 出于对时效性与安全性的考虑还在加密字段中加入了时间戳, 在加密与解密的两次结果中时间间隔过长也应视为异常信息.

当控制器第一次注册时 (即服务器内没有该控制器记录) 视为新入网控制器. 中央服务器会为其分配一个 ID 响应给控制器, 同时在中央数据库生成连接记录, 并将该控制器加入心跳监听序列中.

当注册完成后控制器陆续会将它所在内网中新加入的 ZigBee 节点信息发送至服务器, 此时消息数据段中包括控制器 ID 节点 GlobalID 和节点信息. 在服务

器将新增 ZigBee 节点的数据同步完成后该网络内的节点即可正常使用.

顺利完成第一次注册后控制器会在本地持久化控制器 ID, 当控制器重新启动再次进行注册任务时会携带控制器 ID 表明身份重新激活网络连接. 当控制器各项信息需要更新时便可使用重新注册的方式.

3.3 控制器与中央服务器之间的心跳机制

控制器注册完成之后会周期性的向中央服务器发送心跳包, 服务器会集中管理并监听所有 ZigBee 网络的心跳[12]. 本文的设计中使用了 redis [13,14]中 SortedSet 结构集中管理心跳记录, 成员分数为最近一次心跳的时间戳. 当集合中成员数量大于 64 时 SortedSet 同时使用了 Hash 和 Skiplist[15]两种设计实现, 其范围查找操作复杂度一般为 O(log(*N*)+*M*) (*N* 为有序集的基数, *M* 为被结果集的基数). 使用范围查找在集合中成员分数小于当前时间戳与约定超时时间的差所产生的结果集即可认为是心跳已超时的连接. 服务端将会对结果集内的控制器发送存活确认消息, 当消息响应异常时可视为连接中断, 更改控制器及属于该控制器网络中所有节点的连接状态并执行相应的离线异常处理机制.

3.4 人机接口设计思路

系统内针对用户的查询与控制操作都通过访问中央服务器所提供的 Web 服务实现. 用户可通过使用浏览器或手机 app 等多种互联网手段向服务器发送请求, 服务端最终会将用户的行为翻译成节点 GlobalID 与操作码的形式. 通过 GlobalID 查询数据库, 如果是控制信息将获取节点所在内网的 IP 地址, 再由服务端与指定的 ZigBee 网络进行通讯.

4 简单测试系统的搭建

为验证系统可行性, 利用有限的硬件资源搭建如下测试系统:

系统中央服务器采用阿里云轻量级应用服务器: 1 核 CPU、2 GB 内存、40 GB 固态硬盘、峰值带宽 1 Mbps、操作系统为 CentOS 7.3 版本、提供独立公网 IP 与域名解析功能. 在服务器内使用 Redis 3.0 搭配 MySQL 5.7 版本作为系统中央数据库. 后端代码主要由 Java 语言编写, 所有针对处理用户 http 请求的服务均使用了 Tomcat 服务器. 网站的入口使用 Nginx 作为反向代理服务器用于匹配并分发不同类型的请求.

内网控制器由 Java 语言编写, 并只依赖于 Java 虚拟机, 便于移植. 为验证存在多个内网控制器时系统的运行状态, 使用 VMware 启动 10 台虚拟机作为内网控制器, 运行环境为 CentOS7.3, 利用花生壳内网穿透、端口映射软件将 10 台虚拟机与公网相连. 个人电脑与协调器通过串口相连, 为解决多个虚拟机无法同时访问一个串口的问题, 编写脚本进行串口收发再与虚拟机进行通讯起到模拟相连的作用. 虽然多个控制器是与同一个 ZigBee 网络相连, 但由于相互之间网络地址不同, 数据也不同步, 即可视为多个网络.

系统中协调器与终端均使用 CC2530 芯片, 协议栈版本为 TI-Z-Stack 2007. 网络内有一个协调器搭配 37 枚终端, 部分终端搭载了超声波传感器或温湿度传感器.

在整个测试过程中, 系统能够自行处理任意 ZigBee 节点层面和内网控制器层面的接入与离线情况, 运行稳定鲁棒性强. 通过访问中

央服务器可实时查询任意节点上传感器的测量值, 针对不同内网之间实际相同节点的数据查询极少出现不同步状况. 通过 http 请求的方式对节点上 LED 小灯的控制信息在当前测试环境下操作延迟一般在 3 秒内即可完成响应, 当对于任意一个网络内的节点下达控制信息后, 其余 9 个网络中的节点信息均能在不超过两个节点心跳的时间内同步至中央服务器.

5 结束语

本文提出一种通过搭建互联网服务器的方式来集中管理多个 ZigBee 网络的设计方案. 所有节点的数据集中存储便于访问, 同时可对系统内任意节点实现精准控制与状态信息的获取. 各个 ZigBee 网络之间的设计不存在耦合, 在不同业务背景下应用时不需要更改底层实现, 系统扩展性强. 接下来的工作中需重点关注 ZigBee 网络中对控制指令执行的可靠性问题, 完善各类异常处理机制. 并针对系统各个模块的性能进行更多的测试工作.

参考文献

1 姜仲, 刘丹. ZigBee 技术与实训教程--基于 CC2530 的无线传感网技术. 北京: 清华大学出版社, 2014. 5.

2 任智, 李鹏翔, 姚玉坤, 等. 基于分段的 ZigBee 网络按需可扩展地址分配算法. 通信学报, 2012, 33(5): 131–137. [doi: 10.3969/j.issn.1000-436X.2012.05.017]

3 熊茂华, 熊昕. 无线传感器网络技术及应用. 西安: 西安电子科技大学出版社, 2014.
4 董浩. ZigBee 路由算法的研究及其在数据采集系统中的应用[硕士学位论文]. 银川: 宁夏大学, 2017.
5 杜力凯, 张灵, 陈云华. 一种改进的 ZigBee 路由优化算法. 计算机科学, 2015, 32(5): 153–156, 168.
6 陈海明, 崔莉. 面向服务的物联网软件体系结构设计与模型检测. 计算机学报, 2016, 39(5): 853–871.
7 陈康贤. 大型分布式网站架构设计与实践. 北京: 电子工业出版社, 2014.
8 Membrey P, Hows D, Plugge E. 实用负载均衡技术. 武海峰, 陈晓亮, 译. 北京: 人民邮电出版社, 2013.
9 QST 青软实训. ZigBee 技术开发--Z-Stack 协议栈原理及应用. 北京: 清华大学出版社, 2016.
10 Date CJ. 数据库系统导论. 孟小峰, 王珊, 姜芳艽, 等译. 北京: 机械工业出版社, 2007.
11 Schwartz B, Zaitsev P, Tkachenko V. 高性能 MySQL. 宁海元, 周振兴, 彭立勋, 等译. 3 版. 北京: 电子工业出版社出版, 2013.
12 高爱莲, 刘增磊, 刘中艳. 高可用集群系统的研究. 信息系统工程, 2016, (10): 127–129. [doi: 10.3969/j.issn.1001-2362.2016.10.089]
13 Redis. Redis-doc. https://redis.io/documentation/.
14 申德荣, 于戈, 王习特, 等. 支持大数据管理的 NoSQL 系统研究综述. 软件学报, 2013, 24(8): 1786–1803. [doi: 10.3724/SP.J.1001.2013.04416]
15 Sedgewick R, Wayne K. 算法. 谢路云, 译. 4 版. 北京: 人民邮电出版社, 2012.