中文摘要

摘要：目前，无线传感器网络链路传输技术多是基于 IEEE802.15.4 协议的，难免造成传输速率低、丢包严重等问题，其应用因此也受到限制。异构无线传感器网络可以将多种链路传输技术有效地融合，为用户提供更优质的服务。

论文的工作依托于 CNGI“下一代互联网技术实现智能建筑节能的应用示范”项目，设计并实现了异构无线传感器网络多链路传输技术。

论文首先综述了异构无线传感器网络的研究背景及发展现状，分析了现有异构无线传感器网络中链路传输技术存在的问题。基于 IEEE802.15.4 和 IEEE802.11g 协议，论文提出一种异构无线传感器网络结构。在监控区域内布置高端节点与低端节点。高端节点选用 ARM 节点，提供两种无线接口；低端节点选用单片机节点，无线接口速率相对较低。通过对两种节点和 MAC 层的 IEEE802.15.4 和 IEEE802.11g 协议以及相应的路由协议进行分析，论文提出了多链路传输技术。该技术可以根据信息类型选择合适的链路传输，提高整个网络的性能，优化用户的服务体验。

然后针对异构无线传感器网络结构，论文在 ARM 节点上移植了软件系统平台。在系统平台上设计并实现了多链路传输系统，分别对初始化模块、任务调度模块、数据传输模块、数据采集模块、多链路选择模块和图像传输模块的设计思想和技术实现流程进行详细的分析。论文借助 Linux 内核实现高端 ARM 节点的基本功能，并实现多链路选择和图像数据的采集传输。

最后在实际环境中搭建异构无线传感器网络，测试了两种无线链路上网络的连通性，并通过红外-图像监控的实验验证了多链路传输技术。测试结果表明，多链路传输技术可以使 ARM 节点接入两种无线链路的网络进行通信。通过对数据的多链路选择，数据可以在两种无线链路上传输。该系统可以应用于图像监控的实际场景中。

关键词：异构无线传感器网络；多链路传输技术；IEEE802.15.4；IEEE802.11g
分类号：TP212.09；TN915.04

ABSTRACT

ABSTRACT: Currently, IEEE802.15.4 is mostly used as link transimission technique in wireless sensor network. It inevitably resultes in some problems like limited rate and serious data loss. Application is also restricted. Heterogenous WSN can integrate many link transmission technologies effectively to provide users with better service.

The paper is supported by CNGI' Applications of Next Generation Technology in energy-saving of Intelligent Building' program. We design and implement the multi-link transmission technique of heterogenous WSN.

Firstly, we review background and progress of research about WSN and heterogenous WSN, and analyze existing problems of link transmission techniques. Based on the links of IEEE802.15.4 and IEEE802.11g, we propose the structure of heterogenous WSN. High-end and low-end nodes are arranged in the area. High-end node uses ARM node with two interfaces, while low-end nodes have one. Though analyzation of nodes, MAC and routing protocols, we put forward multi-link transmission technique, which can select right link for different informations. It can improve performance of the network and optimize user's service experiences.

Due to the architecture, we transplant system platform. Based on the playform, we design and implement multi-link transmission system. And then we realize implement initialization, scheduling, data transfer, data acquisition, multi-link selection and image transfer module. We implement basic functions, multi-link selection, image acquisition and transmission, based on system platform of ARM nodes.

Finally building heterogeneous WSN environment, we test the network connectivity of wireless links, and verify the system through the infra-red and image mornitoring. The results show that multi-link transmission system can make ARM nodes access to the two networks to communicate. Based on multi-link selection, data can be transmissed on two kinds of links and returned to server. This system can be applied to image monitoring of actual situations.

KEYWORDS: Heterogenous WSN; Multi-link Transmission Technique; IEEE802.15.4; IEEE802.11g

CLASSNO: TP212.09; TN915.04

致谢

本论文的工作是在我的导师周华春教授的悉心指导下完成的，周华春教授严谨的治学态度和科学的工作方法给了我极大的帮助和影响。在此衷心感谢两年来周华春老师对我的关心和指导。

秦雅娟教授和高德云副教授悉心指导我们完成了实验室的科研工作，在学习上和生活上都给予了我很大的关心和帮助，在此向秦雅娟老师和高德云老师表示衷心的谢意。

牛延超博士、郑涛博士和梁露露博士对于我的科研工作和论文都提出了许多的宝贵意见，在此表示衷心的感谢。

在实验室工作及撰写论文期间，段俊奇、林一多、田洪强、方然、元男、崔英等同学对我论文中的工作给予了热情帮助，在此向他们表达我的感激之情。

另外也感谢家人，他们的理解和支持使我能够在学校专心完成我的学业。

1 绪论

本章介绍无线传感器网络及异构无线传感器网络的研究背景、国内外现状以及选题意义，并概括了论文的主要工作和结构组织。

1.1 研究背景

近年来，无线传感器网络[1]（Wireless Sensor Networks，WSN）得到越来越广泛的关注，特别是计算机网络、微电子技术和无线通信技术的的成熟更促进了无线传感器网络的发展。无线传感器网络是由大量的微型节点散布在监测区域内，通过无线通信的方式自组织而成的网络[2]。无线传感器网络结构如图 1-1 所示。这些微型节点具有体积小、价格低、处理和计算能力受限等特点。传感器节点可以感知、测量并采集监测区域内的感知对象信息，信息经过多跳到达汇聚节点（网关），汇聚节点将信息返回管理节点（服务器）。远程控制端或手机终端便可以通过 Internet 或其他网络，访问管理节点，获取数据或者实现对远程节点的控制。因此拉近了人与物理世界的距离。

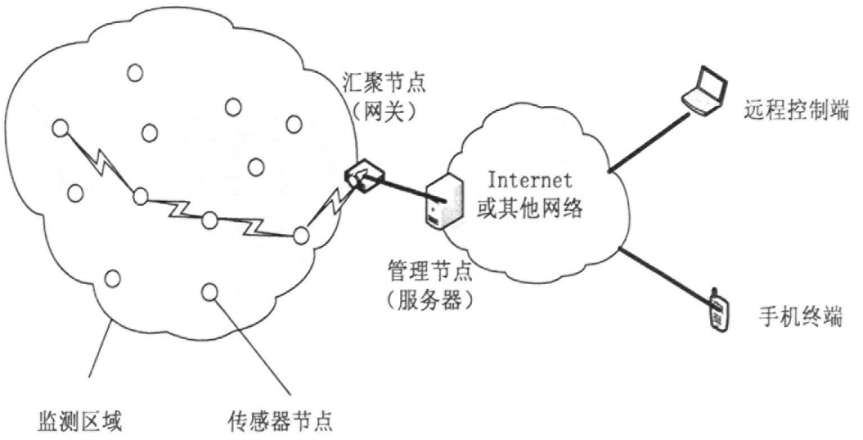

图 1-1 无线传感器网络结构

Fig.1-1.Structure of Wireless Sensor Network

传感器节点是低能耗设备，配置有一个或多个传感器、处理器、存储器、电源、无线通信模块等。各种类型的传感器如光传感器、温度传感器、化学传感器等都可以配备到传感器节点上用来测量环境参数。由于传感器节点的存储空间有限，加之传感器节点通常置于人类难于进入的区域，因此需要采用无线通信的方

式传输数据。传感器节点的电源主要由电池供应，在环境条件允许的情况下，还可以在节点上添加太阳能板等对节点供电。

无线传感器网络在很多的场景下，都发挥了其潜在的应用价值[3]。例如军事目标的跟踪与监测、自然灾害的预警、医疗健康监测、智能化家居等。无线传感器网络的应用发展正关系到国家安全、经济建设等重要方面[4]，吸引了各国政府部门、企业以及研究机构投入大量的资金，极大的推进了无线传感器网络的发展，使之成为一种“无处不在”信息技术[5]，深入到国家的各种领域。

不同于传统的网络，无线传感器网络存在资源受限和设计受限的问题。资源受限包括能量有限、通信范围小、带宽低、以及各个节点的存储和处理能力有限。而设计受限指的是无线传感网络的设计对于应用场景具有很强的依赖性，是完全基于监测环境的。监测环境决定了无线传感器网络的规模、部署方案和网络拓扑，环境条件同样可以影响到节点直接的通信，进而影响网络的连接性。

无线传感器网络的研究方向在于如何在上述的限制条件下提出创新的设计思想、改进已有的协议、构建新型应用场景和开发新的算法等。前期的研究大多数集中在各节点具有相同硬件和软件的网络环境中，这种同构的无线传感器网络结构之所以有很大的吸引性，是因为即使个别节点出现故障失效后，网络仍然能够迅速恢复通信[6]。然而，随着应用需求的增加，异构无线传感器网络[7]逐渐兴起，特别是在实际的应用场景中，这是因为异构无线传感器网络可以提高网络的可靠性、生命周期、传输效率等，同时并没有明显的增加能耗。

异构无线传感器网络的提出旨在将现有无线传感器网络中各种节点硬件、网络协议等有效地融合[6]。无线传感器网络中 IEEE802.15.4 协议的应用、节点硬件等相关技术的发展已相对成熟。另外，IPv6 协议的引入，借助其广泛的地址空间等优势，为无线传感器网络的大规模节点部署提供了许多益处。同时，IEEE802.11 协议[8]、蓝牙协议等作为短距离无线传输[8]中的重要协议，在无线传感器网络中也有一定的应用。因此如何最大地发挥各个因素的优势、如何合理地部署网络都成为了异构无线传感器网络深入研究的方向。

1.2 国内外研究现状

无线传感器网络作为信息产业中计算、通信和传感器三大领域相结合的产物，受到了各国高度关注。WSN 的研究始于 20 世纪 70 年代末期，应用需求的不断增多推动了无线传感网络的发展，同时也带来了很多成果。研究领域涉及到无线传感器网络的各个方面：传感器技术、路由协议、能量优化、传输控制协议、数据融合、多媒体传感技术[9]等。

2003 年，美国科学基金会（NSF）专门制定了 WSN 研究计划[2]，用于研究 WSN 相关理论和技术。美国也极其重视无线传感器网络在军事中的应用：美国陆军的“灵巧传感器网络通信”计划、美国海军的“协同作战能力”（CEC）[10]、国防部在 C4ISR 基础上提出的 C4KISR 计划等都强调了无线传感器网络研究的重要性，利用 WSN 加强战地信息的收集。在民用领域，各国的大学和研究机构也开展了大量 WSN 相关的研究。CodeBlue[11]平台由美国哈佛大学与其他研究单位共同开发，利用 WSN 技术对病患者达到医疗监护的目的，是无线传感器网络在医疗领域应用的典范。2005 年旧金山地区引进 LISA[12]（Lusora Intelligent Sensory Achitecture）系统，主要融合了 Zigbee 技术，用于老年人与病患者的看护。

异构无线传感器网络的研究也在不断的发展和进步中。英特尔公司在其制造平台上应用无线传感器网络来监测机械设备的状态[6]。附着在制造平台上的管道和发动机上的传感器节点可能需要电源供电，而采用电池供电的传感器节点则可以降低安装所需的耗费和复杂性。这类低端传感器节点需要尽可能地节约能耗，因此只具有最简单的计算和通信能力。其他节点放置在可以使用标准插座的地方，并且配有高速微处理器及具有高带宽、长距离的无线通信收发器。高端传感器节点可以进行网内数据处理和长期存储，并能够接入建筑物内的其他网络，如以太网或 802.11 网络。英特尔公司率先将异构无线传感器网络投入到实际应用中，对其日后的发展具有深远的影响。

Millennial Net[6]的研究人员在假设能量和通信能力异构的前提下，开发出异构无线传感器网络的硬件和软件结构。文献[8]和[13]则提出了多链路传输技术的应用实例，即在无线传感器网络中使用异构节点进行数据的处理和传输。

国内的科研单位和学校等也开展了异构无线传感器网络方面的研究。南京大学等提出了异构无线传感器网络的应用平台，包含高端节点和传感节点，高端节点主要起到了类似于网关的功能，而低端节点则负责数据的感应和收集。文献[14]还提出基于 GPRS 与 Zigbee 的无线远程监控网络的研究与应用，这些都可以作为异构无线传感器网络研究的一个方向。

综上，目前异构无线传感器网络结构如图 1-2 所示。低端节点先通过 Zigbee 将数据发送到高端节点，高端节点对数据进行处理后，经过 GPRS 或蓝牙等网络将数据返回服务器，控制终端通过路由器可以获取服务器信息，并对网络进行控制。

根据异构无线传感器网络的发展现状，异构无线传感器网络目前存在的问题如下：

1）通用性差

现有的大多数异构无线传感器网络都是针对特定的应用场景而设计搭建的，加

之无线传感器网络对环境的依赖性较强，往往无法重复利用在其他场景下。

2）部署复杂

异构无线传感器网络的网络结构都处在一个平面上，每种节点仅负责数据采集、处理或传输中的一个功能，大大增加了网络节点部署的复杂性。

3）高端节点功能单一

高端节点仅仅提供了接入不同网络的功能，类似于多接口网关。高端节点中的高级处理器没有得到充分的利用。同时，也不利于在已有的无线传感器网络的基础上添加多媒体等信息的采集和传输。导致无法体现异构网络的优势。

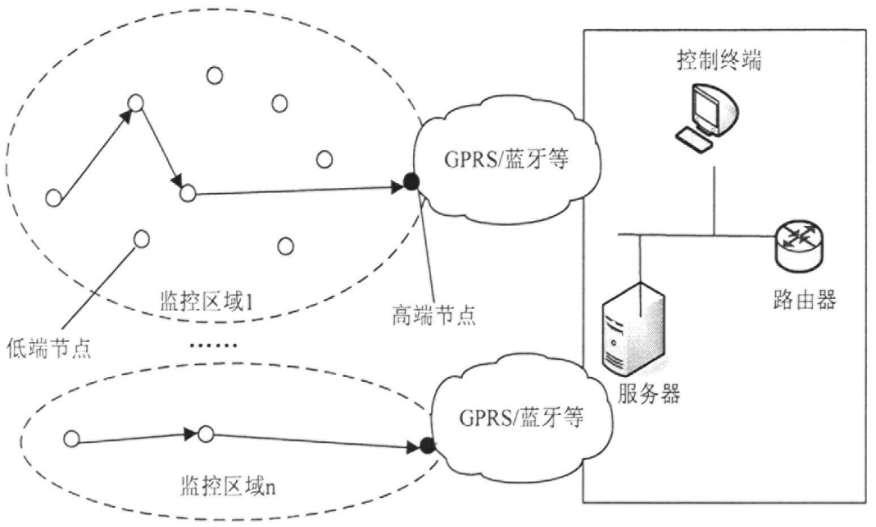

图 1-2 常见异构无线传感器网络体系结构

Fig.1-2.Common Structure of Heterogeneous Wireless Sensor Network

1.3 选题意义

本论文是 CNGI“中国下一代互联网示范工程”项目下的子项目“下一代互联网技术实现智能建筑节能的应用示范”的一部分。这是由北京交通大学下一代互联网互联设备国家工程实验室负责的，旨在将下一代互联网技术引入建筑物中，加速智能化和节能化。在建筑物中部署异构无线传感器网络，保证用户能够随时获得该建筑的各类信息，温度、湿度，甚至是更直观的图像信息等，并可以对建筑内的灯、空调等设备进行远程控制，因此充分体现了建筑楼宇的智能化。所以异构无线传感器网络的研究具有重要的实际意义。

异构无线传感器网络拥有广泛的应用前景，因此其关键技术的研究不仅具有重要的科学研究意义，也会为我国信息领域的发展带来巨大的经济利益。

随着异构的概念深入到无线网络的各个领域，无线传感器网络也可以通过异

构方式来解决现有存在的一些问题。异构无线传感器网络可以兼容已有无线传感器网络的各种关键技术，然而也正是因为这种兼容特性，决定了传统技术无法直接应用于此类型网络。在异构无线传感器网络中，由于节点的计算能力、存储能力以及通信能力都有差异，因此网络采用多种网络协议，如何充分的发挥不同协议的优势，就需要设计出合理的网络架构和多链路传输机制。

IEEE802.15.4 作为现有大多数无线传感器网络的基本 MAC 层协议，满足了无线传感器网络对组网能力、可靠性、节能性和环境条件的要求。然而，随着无线传感器网络应用的日益广泛，用户对信息类型、数据速率、服务质量的要求越来越高，单靠 IEEE802.15.4 协议，对于多媒体流[9]等大量数据、远距离传输、或者网络拥塞等情况，无线传感器网络则无法应对。

IEEE802.11g 具有高数据率、远距离等特点，将 IEEE802.11g 引入 WSN 的优势显而易见。通过 IEEE802.11g，用户能够获得接近于 54Mbit/s 以太网的性能、吞吐率及可用性。IEEE802.11g 的其他一些机制如速率调节机制等都保证了传输的高速性和连接的稳定性。因此在处理多媒体数据和紧急类型数据时，IEEE802.11g 就体现出其优势所在。

本论文提出了一种基于上述两种协议标准下的无线链路提出了异构无线传感器网络多链路传输技术。网络中的高端节点与低端节点一起布置在监控区域内，高端节点具有两种无线接口协议，而低端节点仅具备低带宽的无线链路接口。高端节点是实现多链路传输技术的关键，可以根据不同信息类型选择合适的链路传输，以提高整个网络的性能，优化用户的服务体验。

1.4 主要工作及论文结构

本论文选用目前无线传感器网络常用的无线通信协议标准 IEEE802.15.4 和 IEEE802.11g 作为异构无线传感器网络的底层协议，提出了异构无线传感器网络的结构。针对这种网络结构，深入分析了论文涉及到的两种类型的节点：单片机节点和 ARM 节点，以及节点上不同的网络协议：MAC 层上的 IEEE802.15.4 协议和 IEEE802.11g 协议。深入分析比较了 IEEE802.15.4 和 IEEE802.11g 协议的无线链路特性。相应地，还介绍了节点上的两种路由协议：MSRP 路由协议和 AODV 路由协议。然后论文针对这种异构无线传感器网络结构，移植了 ARM 节点上的系统软件平台，并在该平台上设计实现了节点多链路传输系统，分别对初始化模块、任务调度模块、数据传输模块、数据采集模块、多链路选择模块和图像传输转换模块的设计方案及技术实现进行详细的阐述。最终通过在实际环境中的测试，验证了所设计多链路传输机制的可行性。论文的主要工作包括：

1) 介绍了无线传感器网络的发展现状，并分析了目前异构无线传感器网络存在的问题。
2) 分析了异构无线传感器网络的特性，选择IEEE802.15.4协议和IEEE802.11g协议作为异构无线传感器网络的MAC层协议，提出一种异构无线传感器网络结构。
3) 针对上述异构无线传感器网络，并根据节点的不同功能，提出了多链路传输系统的设计方案和实现流程。
4) 搭建了实际环境并对上述方案进行了测试和验证。
5) 对上述方案进行了总结并对下一步工作进行了展望。

论文的结构为：

第一章 绪论：阐述论文研究背景、国内外研究现状、选题的意义及论文的主要工作。

第二章 异构无线传感器网络：提出了一种异构无线传感器网络结构，并详细阐述其中不同类型的节点、MAC层协议以及路由协议。

第三章 异构无线传感器网络多链路传输系统的实现：移植ARM节点的系统软件平台，对多链路传输技术进行模块化设计，并给出了每个模块具体实现流程。

第四章 系统测试：在实际环境中搭建测试平台，通过测试验证设计方案的可行性。

第五章 总结与展望：总结本论文工作，并对下一步的工作进行展望。

2 异构无线传感器网络

本章介绍异构无线传感器网络中的相关技术。首先提出了异构无线传感器网络的结构。其次，根据该网络结构，详细分析了网络中单片机节点和 ARM 节点的硬件特性。然后阐述了异构无线传感器网络的协议体系，针对 MAC 层协议和路由协议的异构，分别介绍了 IEEE802.15.4 和 IEEE802.11 协议，以及相对应 MSRP 和 AODV 路由协议。

2.1 异构无线传感器网络结构

异构无线传感器网络由高端节点、低端节点、网关组成。其网络结构可以划分为传感网络和覆盖网络两层。异构无线传感器网络结构如图 2-1 所示。异构无线传感器网络中的高端节点 3、4 和 5 除了作为传感网络中的节点之外，还与网关之间还形成了覆盖网络，从而形成了异构无线传感器网分层的结构。

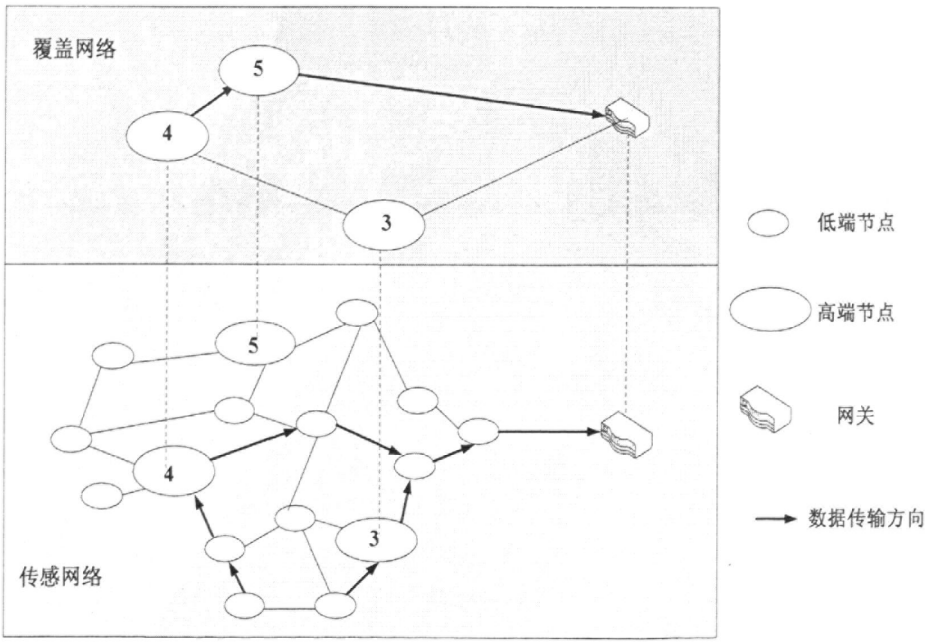

图 2-1 异构无线传感器网络结构

Fig.2-1. Structure of Heterogeneous Wireless Sensor Network

上层的覆盖网络允许高端节点之间，高端节点与网关之间的高带宽和高可靠性的通信。覆盖网络可以是 Ad-Hoc 网络，也可以是现有的有线或无线网络，所以可以利用已有的相对应的路由协议。本论文中上层的覆盖网络是 IEEE802.11g 网络，路由协议为 AODV 协议。

下层的传感网络类似于传统的无线传感器网络，不同的是加入了高端节点，这就要求高端节点除了具有高带宽的无线协议接口外，还需要配备低带宽的无线协议接口，以便与低端节点之间实现通信。论文的传感网络采用的是 IEEE802.15.4 网络，路由协议为 MSRP 协议。

这种网络结构可以很大程度上提高包的成功传输率。在监测区域内布置好高端节点的位置，节点就可以选择通过传感网络中的多跳将数据传送到网关，还是通过覆盖网络的较少的几跳返回数据。

论文的低端节点选用单片机节点，高端节点选用 ARM 节点。

2.2 异构无线传感器网络节点

传感器节点由能量供应模块、传感器模块、处理器模块和无线通信模块组成，如图 2-2 所示。能量供应模块为传感器节点提供运行所需的能量，通常采用便携性电池，也可以采用电源或太阳能电池。传感器模块负责监测区域内信息的传感、采集和数据的转换；处理器模块负责控制传感器节点的所有工作和功能，用于存储和处理各种数据；无线通信模块负责无线信号的收发和处理，收发器将数据交给 MAC 层以及之上的网络协议栈处理，使得能够和其他节点进行通信，交换控制消息和传输数据；

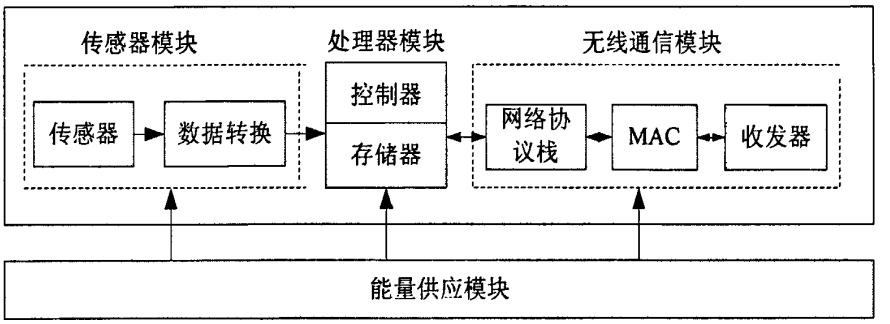

图 2-2 传感器节点体系结构

Fig2-2. Structure of Sensor Node

异构无线传感器网络的节点同样具备上述传感器节点的基本体系结构，但是由于功能需求，异构网络中的节点各有自身的特点。根据异构无线传感器网络中节点所具备的优势资源的类型，可以将节点方面的异构分为三种类型：能量异构（Energy Heterogeneity）、链路异构（Link Heterogeneity）和计算能力异构（Computational Heterogeneity）。

能量异构[15]可以理解为节点能量来源的不同造成节点的异构性。采用便携电池供电的节点，能量存储空间较小，必然对节能的要求更高，而且在部署的时候

还应该考虑到电池更换的问题。而选用固定电源的节点，则无需过多的考虑能量供应的问题，但另一方面，这种节点的部署位置也受到一定的限制。还有一种就是使用太阳能等新型能源的电池，同时解决了上述两类型节点的问题，即保证了能量的持续供应，也避免了部署节点时对于电源位置方面的顾虑。链路异构指的是节点具有不同的网络链路，如 802.11 无线网络链路、802.15.4 无线链路、802.3 有线链路等。节点上可能配备有上述的一种或几种网络接口。计算能力异构[16]则主要指的是节点处理器的不同。三种类型的异构之间相互作用与关联。计算能力异构和能量异构在一定程度上决定了节点的链路异构。

计算能力异构是节点异构中最重要的一个因素。处理器是无线传感器节点的计算核心，所有的设备控制、任务调度、能量计算、功能协调、通信协议、数据整合、数据转储的功能都需要处理器的支持才能够完成，所以处理器的选择对于传感器节点至关重要。在节点处理器方面，目前的传感器网络节点一般选择单片机作为处理器。单片机具有体积小、功率小等主要特点，同时，单片机也具有处理能力弱、存储空间小等致命性弱点。这在一定程度上会导致基于单片机的传感器节点，仅适合采集和处理简单类型数据，而不适合处理多媒体数据，因此将单片机节点作为低端节点。高端节点则可以采用 ARM 节点。ARM[17]作为一种嵌入式系统处理器，以高性能、低功耗、低成本、体积小等优点已经成功地广泛应用于无线通信、工业控制、消费类电子产品、网络产品等领域。将 ARM 处理器选做无线传感器网络节点处理器可以从根本上解决以往单片机节点的处理数据慢、存储空间过小、功能单一等问题，进而可以提高网络的性能。可以在 ARM 处理器上方便地加载操作系统也是 ARM 广受欢迎的重要因素之一，因为操作系统能够为用户的开发提供方便的开发平台，因此也可以实现更加丰富的功能。

2.2.1 单片机节点

单片机节点的处理器使用 ATMEL 公司的 ATmega128L[18]处理器。它采用的是低功耗的基于 RISC（Reduced Instruction Set CPU）结构的 8 位微控制器，也是目前 AVR（Automatic Voltage Regulator）系列中功能最强大的单片机。ATmega128L 处理器的特点如下：

- 采用哈佛结构，具有 1MIPS/MHz 的运行处理能力；
- 快速的存取寄存器组、单周期指令系统，使得目标代码的大小和执行效率得以优化；
- 非易失性的程序和数据存储器 ：128K 字节可编程 FLASH，4K 字节的 EEPROM 以及 4K 字节的 SRAM；

- JTAG 测试接口，支持扩展的片上调试；
- 53 个可编程 I/O 口，4 个具有比较功能和 PWM 功能的定时器，两个 USART，TWI，8 通道 10 位 ADC，SPI 串行端口；
- 支持上电复位和可编程的掉电检测；
- 允许通过软件选择时钟频率；
- 6 种睡眠模式: 空闲模式、ADC 噪声抑制模式、省电模式、掉电模式、Standby 模式和扩展的 Standby 模式；

单片机节点的处理器模块由电源模块提供 3.3V 直流电。时钟源由 8MHz 的贴片晶振提供，具有体积小、性能稳定的特点，抗干扰能力较强，IEEE802.15.4 无线通信模块通过 SPI 接口与处理器模块相连。论文采用的单片机节点如图 2-3 所示，由无线通信模块、传感器模块、能量供应模块和基于 ATmega128L 芯片的处理器模块组成。

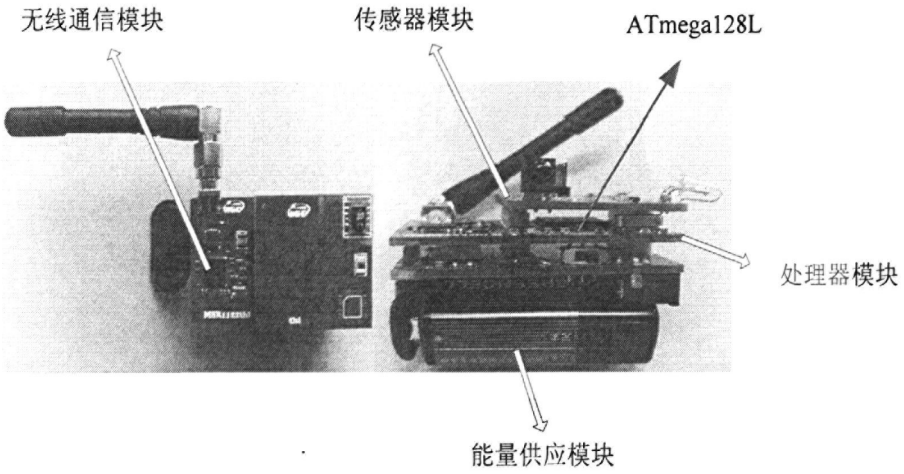

图 2-3 单片机节点

Fig2-3. Single Chip Node

2.2.2 ARM 节点

ARM[17]（Advanced RISC Machines）节点作为异构无线传感器网络中的高端节点，主处理器采用的是 S3C2440[19]。三星公司推出的 16 位/32 位 RISC 微处理器 S3C2440 采用了 ARM920T 的内核，为微型处理器应用提供了低价格、低功耗、高性能的解决方案， S3C2440 的性能特点如下：

- 主频 400MHZ，最高可达 533 MHZ 和高达 1G 的地址空间；
- 独立的 16KB 指令 Cache 和 16KB 数据 Cache；
- 时钟和电源管理采用片上 MPLL 和 UPLL；

- 60 个中断源，包括电平/边沿触发模式的外部中断源；
- 4 通道 16 位具有 PWM 功能的定时器以及 1 通道 16 位内部定时器；
- 24 个外部中断端口和 130 个多功能输入/输出端口均为通用 I/O 口；
- 4 通道的 DMA 控制器，通过触发传输模式可以提高传输速率；
- 16 位看门狗定时器，在定时器溢出时发生中断请求或系统复位；
- 1 通道 IIC 总线接口，支持 8 位串行双向数据传输，数据传输速度可达 100kbit/s，快速模式下可达到 400kbit/s；
- 1 个 USB 主设备（USB Host）接口和一个 USB 从设备（USB Device）接口，允许连接各种 USB；设备，如摄像头、无线网卡、接口转换设备；
- SPI 接口，具有 2*8 位的移位寄存器，支持基于 DMA 或中断模式工作。
- 全性能的 MMU，支持操作系统

处理器之外的存储器，ARM9 具有 64MB 的 NAND Flash 和 2MB 的 NOR Flash，其芯片分别采用的是 K9F1208 和 Am29LV160D。Flash 是一种非易失性存储单元，支持掉电保护，可以通过编程进行电擦写，这些特性决定了它更适用于便携式设备。本论文的 S3C2440 设置为由 NAND Flash 启动引导：采用 4KB 内部缓冲器进行启动引导。

ARM 节点如图 2-4 所示。

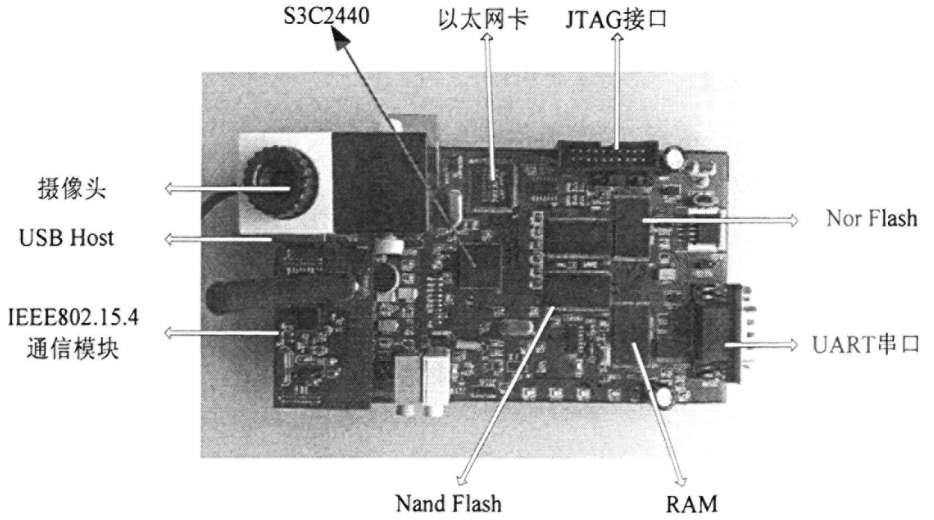

图 2-4 ARM 节点

Fig2-4. ARM Node

2.2.2.1 ARM 节点结构

基于传感器节点基本结构，ARM 节点的结构可以分为：控制模块、存储模块、

电源管理模块、传感模块、以及多种无线通信模块。传感模块可以分为多媒体数据传感和简单型数据传感，无线通信模块可以分为 IEEE802.11g 无线网卡和 IEEE802.15.4 无线射频模块。ARM 节点结构如图 2-5 所示。

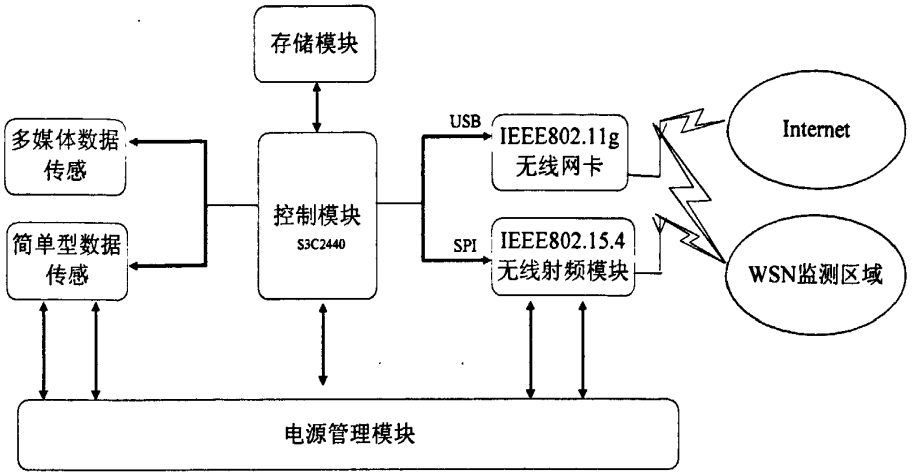

图 2-5 ARM 节点结构

Fig2-5.Structure of ARM Node

2.2.2.2 外围设备

1）IEEE802.15.4 通信模块

IEEE802.15.4 通信模块采用的是 CC2420[20]射频芯片。CC2420 是 Chipcon 公司推出的一款符合 IEEE 802.15.4 标准的射频芯片，工作频带为免授权 2.4G ISM 频带，数据传输速率可达 250kbp/s，支持 CSMA/CA 功能，传输距离最大为 200m，内部集成可用于实现节点测距功能的数字 RSSI 模块、电源监控和信道变换等功能模块，包含硬件 MAC 和 CRC 自动校验处理，具有高达-94dBm 的接收灵度。CC2420 是专门针对无线传感器网络设计的射频通信芯片，能够很好地满足其需求。具有集成度高，体积小，能耗低的特点，通过 SPI 接口与 S3C2440 连接。

2）IEEE802.11 通信模块

本论文中选用华硕的一款型号为 WL-167G[21]的 USB 接口无线网卡作为 IEEE802.11 通信模块。该 USB 无线网卡芯片为 rt73，兼容 IEEE802.11 系列协议标准，包括 IEEE802.11g、IEEE802.11g。工作频段在 2.4G ISM 频带，支持 OFDM、CCK、DQPSK、DBPSK 的数据调制方式。最高传输速率达 54Mbps。天线增益为 1.77dBi，即使在存在其他无线网络、蓝牙和微波设备干扰的情况下，也能保持较高的接收灵敏度。有效工作的传输距离室内为 40 米，室外可达 330 米。WL-167G 无线网卡小巧便携，并且方便安装，直接连接到 ARM 节点的 USB Host 接口使用。

3）图像采集模块

图像采集模块的处理芯片是中星微公司生产的 ZC0301[22]芯片。ZC0301 芯片是一款廉价的 DSP 控制芯片，集成了控制器、图像信号处理器(Image Signal Processor， ISP)、图像子采样光栅、JPEG 编码器以及 USB 设备控制器。图像传感器采用的是美国镁光公司生产的 MI360，具有 30 万像素，通过主控芯片提供的 CMOS 图像传感器总线接口，图像传感器与主控芯片可以实现无缝连接。最终通过 USB Host 与 S3C2440A 连接在一起。

2.3 异构无线传感器网络协议

异构无线传感器网络的网络协议体系结构具有一定的特殊性，可以按照物理层、MAC（Medium Access Control，介质访问控制）层、网络层、传输层和应用层进行分层，但对于不同的节点，其网络协议体系也有差异性。异构无线传感器网络体系结构如图 2-6 所示。

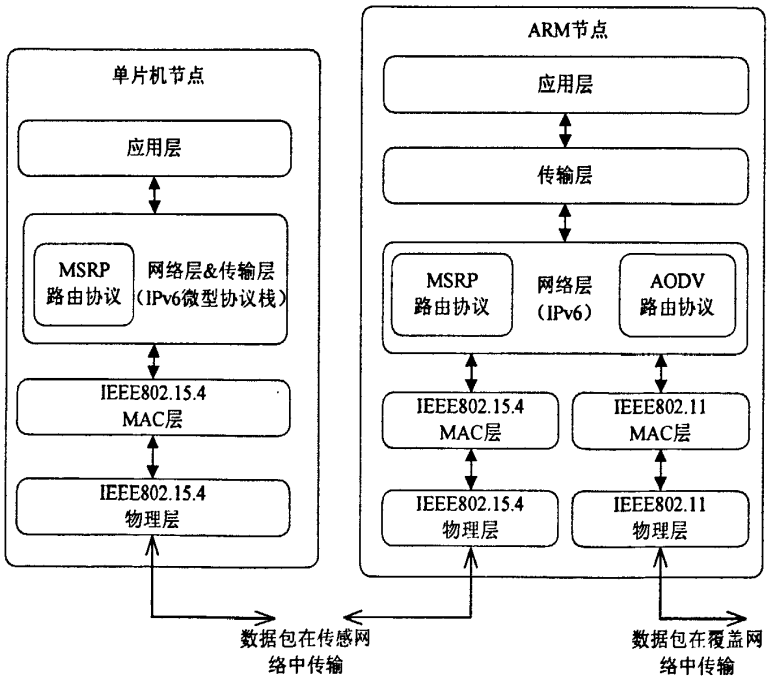

图 2-6 异构无线传感器网络体系结构

Fig2-6.Structure of Heterogeneous Wireless Sensor Networks

物理层和 MAC 层协同工作，为物理设备之间提供的硬件支持[24]和介质访问控制接口。

适配层是因特网工程任务组（IETF）的 6Lowpan 工作组针对无线个人区域网络架构 IPv6 协议提出的工作组草案，它提供 IPv6 协议栈的处理接口，负责将数据

递交到 IEEE 802.15.4 介质访问层和 IEEE 802.15.4 物理层进行封装、调制和发送。

网络层对下层不同的通信协议提供了统一的处理接口，具有很好的兼容性和可扩展性。IPv6 作为下一代互联网的核心网络层协议，具有地址资源丰富、安全性高、移动性好、支持地址自动配置和实时业务等优点。由于单片机节点的硬件资源有限，因而网络层部分采用的是 IPv6 微型协议栈，由北京交通大学下一代互联网互联设备国家工程实验室开发。而对于 ARM 节点，则可以直接利用 Linux 内核的网络协议栈。路由协议根据不同的底层通信协议而选用 MSRP 和 AODV 路由协议，用于传感器网络内数据传输时建立和维护路由。

传输层负责数据的传输与控制，保证了端到端的通信，通过差错控制和流量控制等机制为数据提供了可靠性的传输。

应用层是传感器网络节点提供各种传感应用程序服务的集合，主要包括数据采集、数据处理、数据融合等功能。

下面分别对异构传感器网络的 MAC 层协议和路由协议进行详细阐述。

2.3.1 MAC 层协议

异构无线传感器网络中的单片机节点仅具备 IEEE802.15.4 一种 MAC 协议，而 ARM 节点上实现了 IEEE802.15.4 和 IEEE802.11g 两种协议标准。IEEE802.15.4 和 IEEE802.11g 协议对应的无线链路具有不同的特性，对多链路传输的基础。

2.3.1.1 IEEE802.15.4 协议

IEEE 802.15.4[25]是低速无线个域网媒体访问控制层和物理层标准。2000 年 12 月 IEEE 成立 IEEE802.15.4 工作组，致力于定义一种廉价、便携、移动、低成本、低复杂度、低能耗和低速率的无线连接技术。突出优点是：组网能力强，适用面广，可靠性高，节能性好。特别是实现 IEEE802.15.4 协议标准所需的硬件资源很小，制作工艺简单，大大降低了成本。IEEE802.15.4 协议已经在无线传感器网络中得到了广泛的应用。

IEEE802.15.4 主要制定了物理层和 MAC 协议标准。物理层子层包含射频(RF)模块和物理层控制机制。它定义了两个物理层标准，分别是 2.4GHz、868Mhz、915Mhz 物理层，每个频段提供不同的数据传输速率 250 kb/s、20 kb/s 和 40kb/s；提供自动帧确认和数据校验等功能。特别在 2.4GHz ISM 频段中定义了 16 个信道，每一信道宽 3MHz，信道中心间隔为 5MHz。IEEE 802.15.4 标准的 MAC 子层提供物理信道的访问控制和帧的封装，支持 Beacon（信标）和 Beaconless（无信标）

两种工作模式。Beacon 模式利用超帧结构进行同步，采用基于时隙的 CSMA-CA，使不工作的设备进入低功耗的睡眠状态，有效地节约电能；Beaconless 工作模式则直接使用 CSMA-CA 的机制，用于避免传输碰撞。

IEEE 802.15.4 规定 MAC 帧格式如图 2-7 所示，最大帧长度为 127 字节。MAC 帧由三个基本部分组成：MHR（MAC Header，MAC 头部）、MAC payload（MAC 净荷）和 MFR（MAC Footer，MAC 尾部）。

<table>
<tr><td>Octets: 2</td><td>1</td><td>0/2</td><td>0/2/8</td><td>0/2</td><td>0/2/8</td><td>variable</td><td>2</td></tr>
<tr><td rowspan="2">Frame control</td><td rowspan="2">Sequence number</td><td>Destination PAN identifier</td><td>Destination address</td><td>Source PAN identifier</td><td>Source address</td><td rowspan="2">Frame payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Addressing fields</td></tr>
<tr><td colspan="6">MHR</td><td>MAC Payload</td><td>MFR</td></tr>
</table>

图 2-7 IEEE802.15.4MAC 帧格式

Fig2-7. IEEE802.15.4 MAC frame format

各字段值的表示意思如下：

- Frame control（帧控制域）：16bit 的帧控制域包含了帧类型、安全使能、地址字段、内部 PAN 和其它控制标志位；
- Sequence number（序列号）：标识帧的唯一序列号；
- Destination PAN identifier（目的 PAN ID）：16bit，表示该帧的接收节点；
- Destination address（目的地址）：16 位短地址；
- Source PAN identifier（源 PAN ID）：16 位，唯一表示该帧的发送节点；
- Source address（源地址）：16 位短地址或者 64 位扩展地址；
- Frame payload（帧载荷）：长度由不同类型的帧的不同内容决定；
- FCS：16 位的 ITU-T CRC，可以由 MHR 和 MAC Payload 计算校验和得出。

IEEE 802.15.4 定义了四种帧格式，由帧控制域中的 Frame type 说明，四种帧分别是 Beacon 帧、数据帧、确认帧和 MAC 命令帧。Beacon 帧的主要功能是在网络中的同步作用，在 Beacon 帧中只有源地址信息而没有目的地址信息；数据帧的负载字段包含了上层需要传输的数据；确认帧是针对上次接收的命令帧或数据帧的确认回复；MAC 命令帧用来表示 MAC 层的相关命令，包括连接请求、数据发送请求等。

2.3.1.2 IEEE802.11g 协议

2003 年 7 月 802.11 工作组批准了 IEEE802.11g[26]标准。IEEE802.11g 综合了

IEEE802.11a 和 IEEE802.11b 的优势。它工作在 2.4GHz 频带上，它采用了 OFDM 调制技术实现了高速的数据传输，传输速率达到 54Mbps；在组帧方式上，802.11g 的物理帧结构采用 OFDM/OFDM 方式。IEEE802.11g 的使用范围在室外为 300 米，在办公环境中则最长为 100 米。IEEE 802.11g 无线局域网与以太网的原理很类似，都是采用载波侦听的方式来控制网络中信息的传送，不同的是 802.11g 无线局域网则引进了 CSMA/CA(载波监听多路访问/冲突避免)技术和 RTS/CTS(请求发送/清除发送)技术，从而避免了网络中冲突的发生，可以大幅度提高网络效率。

IEEE 802.11g 规定 MAC 帧格式如图 2-8 所示，最大帧长度为 2346 字节，包含了 MAC 头部、负载以及尾部信心。

<table>
<tr><td>Octets: 2</td><td>2</td><td>6</td><td>6</td><td>6</td><td>2</td><td>6</td><td>2</td><td>0~ 23424</td><td>4</td></tr>
<tr><td rowspan="2">Frame control</td><td rowspan="2">Duration / ID</td><td>Address 1</td><td>Address 2</td><td>Address 2</td><td>Sequence Control</td><td>Address 4</td><td>QoS Control</td><td>Frame body</td><td>FCS</td></tr>
<tr><td colspan="5">Addressing fields</td></tr>
<tr><td colspan="8">MHR</td><td>MAC Payload</td><td>MFR</td></tr>
</table>

图 2-8 IEEE802.11gMAC 帧格式

Fig2-8.IEEE802.11gMAC frame format

- Frame control（帧控制域）：包含了协议版本、帧类型、分段情况、重传域、能量管理域、加密数据域和序号域；
- Duration/ID （持续时间 / 标识）：表明了该帧和它相对应的确认帧将会占用信道的时间；
- Address （地址域）：包括源地址（ SA ）、目的地址（ DA ）、传输工作站地址（ TA ）和接收工作站地址（ RA ）， SA 与 DA 必不可少，后两个只针对跨 BSS 的情况使用；
- Sequence Control （序列控制域）：由代表 MSDU （ MAC Server Data Unit ）或者 MMSDU （ MAC Management Server Data Unit ）的 12 位序列号（ Sequence Number ）和表示 MSDU 和 MMSDU 的每一个片段的编号的 4 位片段号组成（ Fragment Number ）Source address（源地址）：

针对帧的不同功能，可将 802.11g 中的 MAC 帧分为三类。控制帧用于竞争期间的握手通信和正向确认、结束非竞争期等；管理帧主要用于 STA 与 AP 之间协商、关系的控制，如关联、认证、同步等；数据帧用于在竞争期和非竞争期传输数据。Frame Control（帧控制域）中的 Type（类型域）和 Subtype（子类型域）共同指出帧的类型，当 Type 的为 00 时，该帧为管理帧；为 01 时，该帧为控制帧；

为 10 时，该帧为数据帧。

2.3.2 路由协议

传感网络和覆盖网络的路由是相互独立的。因此分别针对底层的 IEEE802.15.4 和 IEEE802.11g 协议标准，采用不同的路由协议 MSRP 和 AODV 路由协议。

2.3.2.1 MSRP 协议

MSRP（Micro Sensor Routing Protocol，微型传感器路由协议）是北京交通大学下一代互联网互联设备国家工程实验室自主开发的路由协议。能够根据无线传感器网络的特点，实现路由功能。

➢ MSRP 的路由建立过程

当节点第一次向目的节点发送数据包时，节点先缓存数据包，通过广播路由请求报文（RREQ），发起路由查询功能。路由请求报文包括目的地址、源地址、路由请求 ID、跳数等。而且每个节点都会维护一个路由请求表用于记录其它节点发来的 RREQ，主要包括路由请求 ID 和源地址。

当中间节点收到 RREQ 时，处理过程如图 2-9 所示。

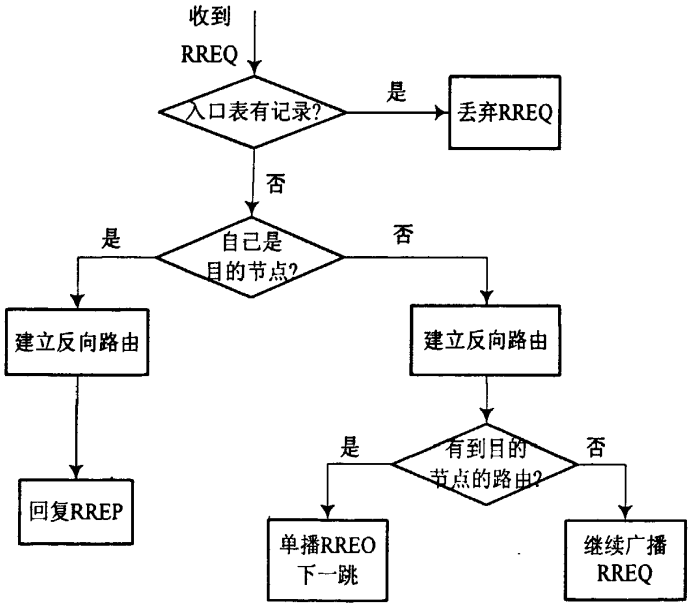

图 2-9 节点收到 RREQ 流程图

Fig2-9. Flow chart of nodes receiving RREQ

中间节点会先查找路由请求表，若以存在该 RREQ 表项，则丢弃 RREQ；否在表中插入新 RREQ 表项。根据新 RREQ 信息，再继续查找路由表，如果存在目

的节点的路由信息，则单播 RREQ 到路由表的下一跳节点；否则继续广播 RREQ，直到目的节点。在将 RREQ 发往目的节点的同时，中间节点建立起到源节点的反向路由，路由回复报文（RREP）将沿着反向路由到达源节点。

当目的节点收到 RREQ 时，缓存该 RREQ 分组消息。目的节点会根据时间 T 内的收到的多条的 RREQ 信息，计算得出链路质量最好的路由进行回复。路由回复报文 RREP 包括源节点、目的节点以及跳数。

中间节点收到 RREP 分组时，建立到目的节点的路由，然后查找到节点的反向路由，转发 RREP。直到源节点收到 RREP，则路由建立就完成。

2.3.2.2 AODV 协议

AODV [27](Ad hoc On demand Distance Vector Routing，无线自组网按需平面距离矢量路由协议）是 IETF MANET 工作组提出的基于 Ad Hoc 网络的一种按需驱动的路由协议。作为一种反应式路由协议，AODV 只有在需要向目的节点发送封包时，才在网络中发起路由查找过程。所以不需要节点维护处于非活动状态的路由信息。

- AODV 的路由建立过程

每个节点通过其他节点定期广播 HELLO 包，确定自己的邻居节点，如图 2-10 所示，节点 1 的邻居节点为节点 2 和 4。当节点 1 第一次向节点 3 发送数据时，就需要新建立一条节点 1 到节点 3 的路由。

节点 1 发出一个 RREQ，由节点 2 和 4 接收到这个 RREQ。则节点 2 和 4 对于 RREQ 有两种处理方式：如果已知到目的节点的路由或者本身为目的节点，它们会向节点 1 返回 RREP；否则的话，就会向他们的邻居节点继续广播 RREQ。如图 2-10 所示，节点 2 具有到节点 3 的路由，因此回复 RREP 给节点 1，而节点 4 没有查找到相关路由，节点 4 会重新广播 RREQ。进而建立了节点 1 到节点 3 的路由。

- AODV 的安装

由于 ARM 节点具有操作系统平台，而且 AODV 也提供了对 ARM 平台的支持，所以 AODV 的移植显得方便了许多。论文选用的 AODV 源码版本是 aodv-uu-0.9.5.tar.gz，安装步骤如下：

1. 将 aodv-uu-0.9.5.tar.gz 后解压，进入该文件夹；
2. 直接对 ARM 平台专用的 Makefile 进行编译；

```
#make arm
```

3. 将生成的 kaodv.ko 驱动程序和 aodvd 可执行文件下载的 ARM 节点的任意文件下，安装驱动，并运行可执行文件，为方便后续开发，可以将 aodvd 改

为后台程序运行，独立于终端控制；

#insmod kaodv.ko

#./aodvd

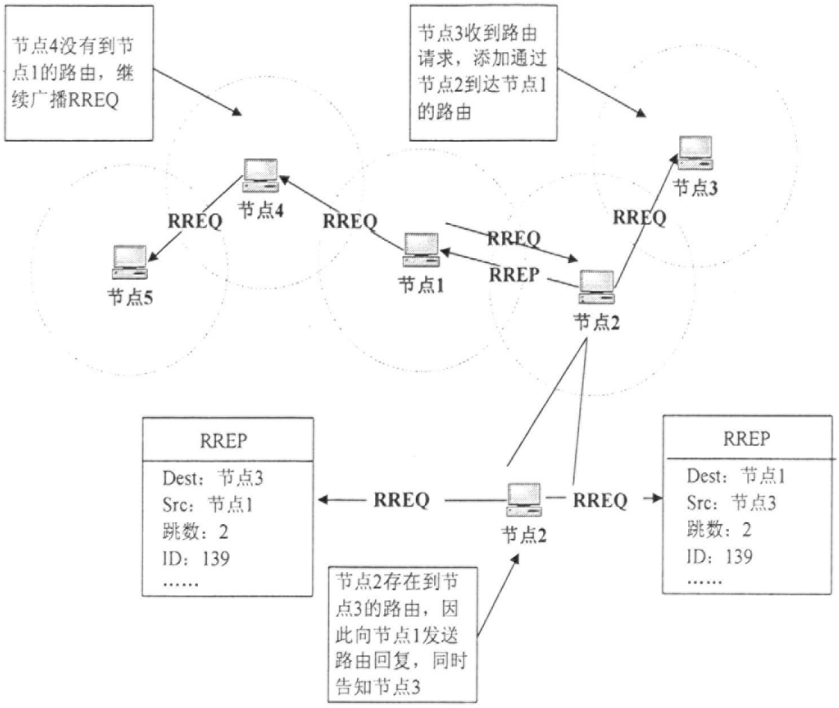

图 2-10 AODV 路由建立过程

Fig2-10.Construct Process of AODV Route

2.4 多链路传输技术

多链路传输技术是异构无线传感器网络中的关键技术，主要研究在网络链路异构的情景下，实现数据在不同链路之间的不间断传输。并且在高带宽的链路上可以传输多媒体数据等信息，也可以分担其他链路的网络负载，进而弥补单一链路的不足之处。

多链路传输技术是基于高端节点实现的。由于高端节点的丰富的硬件资源和强大的处理能力，所以高端节点可以实现多种网络接口。ARM 节点就是通过IEEE802.15.4和IEEE802.11g两种网络接口分别与传感网络和覆盖网络中节点相连通，为数据的传输提供了选择不同链路传输的可能。多链路传输技术如图 2-11 所示。

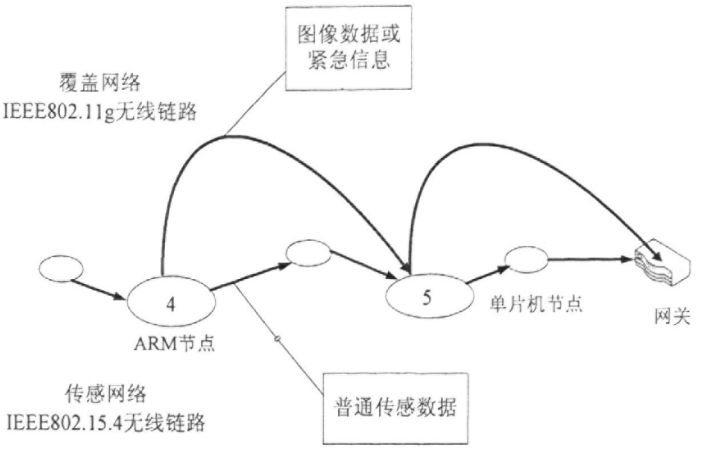

图 2-11 多链路传输技术

Fig.2-11. Multi-link Transmission Techniques

ARM 节点 4 对自己采集的数据以及由其他节点采集的需要节点 4 转发的数据都会进行多链路选择的处理。根据处理结果，数据会通过传感网络发送到网关，或者在覆盖网络上传输，经由另一高端节点 5 到达网关。例如，ARM 节点采集的图像数据会选择在覆盖网络中传输。ARM 节点转发的普通传感数据则还会通过传感网络传输。而对于从低端节点发来的紧急信息，经过 ARM 节点处理之后，会由传感网络进入到覆盖网络传输，尽快地发送到网关。

多链路传输技术实现的关键在于 ARM 节点首先应该保证每个链路的网络连通性。对于 IEEE802.15.4 网络，实验室开发的单片机节点虽然在 IEEE802.15.4 协议的基础上利用 IPv6 协议栈实现了节点的数据传输功能，但是 ARM 节点的操作系统是 Linux，其内核协议栈已经非常完善。因此 IEEE802.15.4 协议之上的数据传输功能需要根据 Linux 的特点重新设计和实现。对于 IEEE802.11g 网络，ARM 节点的 Linux 对 IEEE802.11g 已经提供了相应的支持，只需要完成移植的工作。在 ARM 节点上还要实现图像数据的采集和传输以及多链路选择的功能。最终才能够完整的实现多链路传输技术。

2.5 本章小结

本章对异构无线传感器网络的相关技术进行介绍。首先详细阐述了异构无线传感器网络的网络体系结构。分析了异构无线传感器网络节点的异构类型，针对本论文涉及到的两种节点：单片机节点和 ARM 节点分别进行了分析。在此基础上介绍了两种 MAC 协议 IEEE802.15.4 和 IEEE802.11g，及路由协议 MSRP 和 AODV。最后介绍了异构无线传感器网络的多链路传输技术。

3 异构无线传感器网络多链路传输系统的实现

基于上一章介绍的异构无线传感器网络，论文设计并实现了该网络系统下的多链路传输系统。首先在 ARM 节点上移植了系统软件平台，在该平台上分初始化模块、任务调度模块、数据传输模块、数据采集模块、图像传输模块以及多链路选择模块进行详细设计和技术实现。

3.1 总体设计

异构无线传感器网络的多链路传输在操作系统平台下实现，关系到网络的各个协议层次，从数据的采集、各层包头的封装，到网络层对下层多链路的通用透明化处理，以及数据在两种链路之间的转换。因此按照网络的层次结构，划分初始化模块、任务调度模块、数据传输模块、数据采集模块、图像传输模块以及多链路选择模块。异构无线传感器网络的多链路传输技术的总体设计如图 3-1 所示。

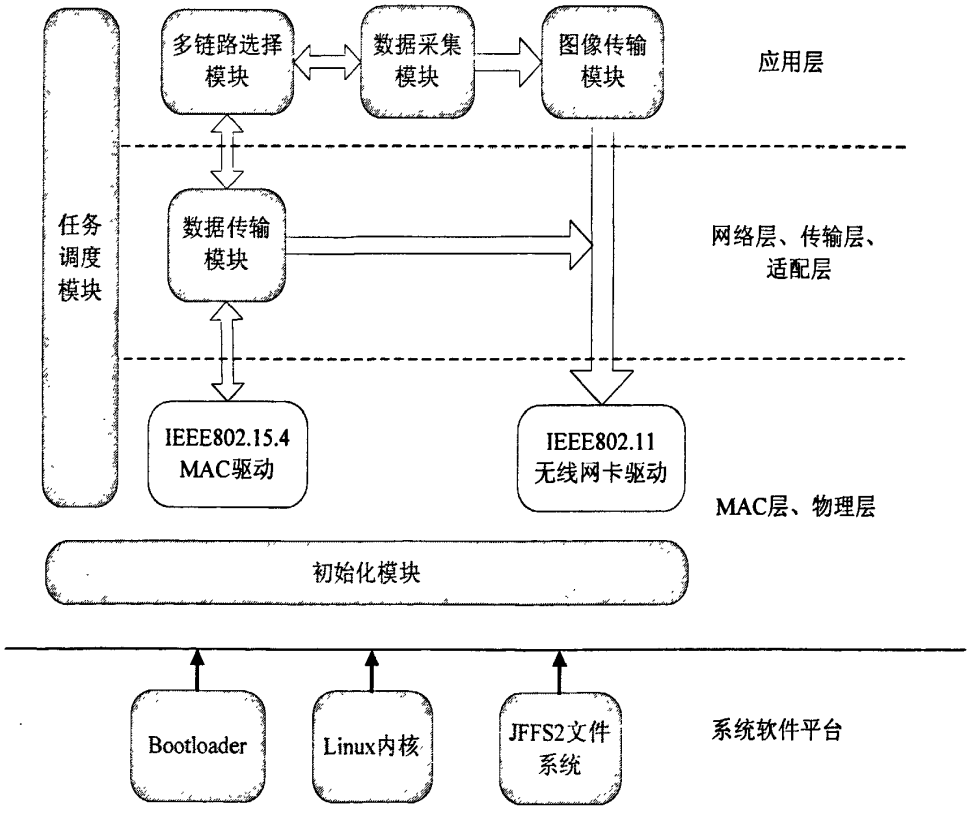

图 3-1 多链路传输技术的总体设计

Fig3-1. Design of Multi-link Transmission Technique

系统软件平台是实现多链路传输系统的基础。操作系统的添加可以为开发提供更好的可扩展性。实验室开发的单片机节点在无操作系统情况下实现了IEEE802.15.4网络节点的基本功能。ARM节点不同于单片机节点，能够为操作系统提供丰富的硬件资源。论文利用操作系统实现了节点基本功能，并且丰富了数据采集模块，添加了图像传输模块和多链路选择模块。完整的系统软件平台包括Bootloader、内核和文件系统。论文选用的是Linux内核和JFFS2文件系统。

IEEE802.15.4MAC驱动由实验室开发完成，对CC2420无线通信模块提供了底层协议支持。IEEE802.11g无线网卡驱动对无线网卡也提供了通信支持。论文的总体设计是在IEEE802.15.4MAC驱动和IEEE802.11g无线网卡驱动的基础上完成的。

初始化模块并不属于网络协议结构中的任何一层，主要是为其他模块和驱动提供初始化的服务。初始化模块分为两部分 IEEE802.15.4 驱动的初始化和IEEE802.11g无线网卡驱动的安装及配置。此外还为其他模块提供了内核定时器、钩子函数、套接字等的初始化。

任务调度模块贯穿了网络的各个层次。利用Linux内核的调度机制，使得繁多的各项任务能够有条理得工作。

数据传输模块是多链路传输技术中的重要模块，既要保证各种数据正常的发送和接收，更重要的是为上层的应用提供统一的接口，底层的多链路对于上层应用是透明的。

协议转换模块仅针对要发送的数据，根据多链路选择模块判断的结果，对需要协议转换的数据进行相应的处理，在传感网络传输的数据就可以在覆盖网络中继续传输。

多链路选择模块分为应用层数据的多链路选择和适配层的多链路选择。应用层和适配层可以对数据包类型字段进行修改。在数据包发送时，进入MAC层处理前，ARM节点会对数据包进行判断，利用网络地址的替换为数据包选择链路。

数据采集模块主要负责各种传感数据的采集，本论文重点介绍了图像数据的采集实现过程。

图像传输模块专门负责图像数据的传输工作，由于多媒体数据的特殊性，因此决定了这类型数据只能在覆盖网络中传输。该模块包括了图像的发送和接收过程的描述。

3.2 系统软件平台

多链路传输系统是在操作系统平台上实现的，因此ARM节点的软件系统平台

是后续开发的基础。论文在 ARM 节点上实现了软件系统平台的移植，即 Boot Loader、操作系统内核和文件系统的移植。

3.2.1 ARM 节点的系统软件平台

ARM 节点的处理芯片 S3C2440 提供的一系列的 MMU、FLASH、DMA 存储器等都为操作系统的移植奠定了良好的硬件平台。ARM 节点中软件系统由下至上分为 Boot Loader、操作系统内核、文件系统和用户应用程序[27]，如图 3-2 所示。

- Boot Loader：操作系统启动前的内核引导程序。用于初始化硬件设备、建立内核空间的映射；
- 操作系统内核：Linux 内核、WinCE 内核等，是 ARM 节点软件系统的核心部分；
- 文件系统：在操作系统中负责管理和存储文件信息的软件结构；
- 用户应用程序：在嵌入式 Linux 下开发的相应程序。

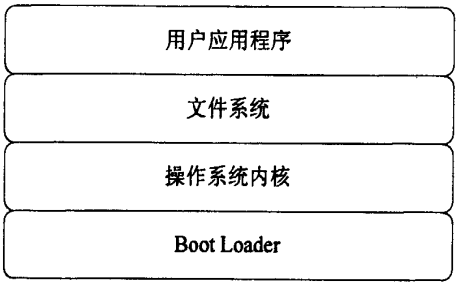

图 3-2 ARM 节点的软件系统

Fig3-2.Software System of ARM Node

ARM 节点软件系统平台的移植主要指的是 Boot Loader、操作系统内核、文件系统的移植。对于操作系统，Linux 操作系统[17]具有免费、开源、稳定、可靠、安全、网络功能强大等优点。Linux 的特点如下：

- Linux 开放源代码，同时还具有大量好用的开发工具，用户可以完全打造出一个适合自己使用的 Linux；
- Linux 的内核小、效率高，运行时所需资源少，非常适用于资源有限的 ARM 处理器；
- Linux 是多用户多任务的操作系统，由于 Linux 的系统调度，允许同时执行多个程序，而且各个程序的运行是互相独立的；
- Linux 具有完善的网络功能，支持大部分标准的网络协议；
- Linux 内核移植方便，操作简单；

嵌入式文件系统的分类主要取决于所依赖的存储介质。非易失性闪速存储器

Flash 具有速度快、成本低、密度大的特点，因此被广泛的应用于嵌入式系统。Flash 存储器可以分为 NOR Flash 和 NAND Flash。总体来说，NOR Flash 的读取速度快、写入速度慢、随机存取快、单片容量小，NAND Flash 则相反。

文件系统与 Flash 之间通过 MTD（Memory Technology Device）建立连接，MTD 为 Flash 提供了很好的驱动支持，同时还对文件系统抽象出统一的接口函数。MTD 与 Flash 和文件系统的关系如图 3-3 所示。

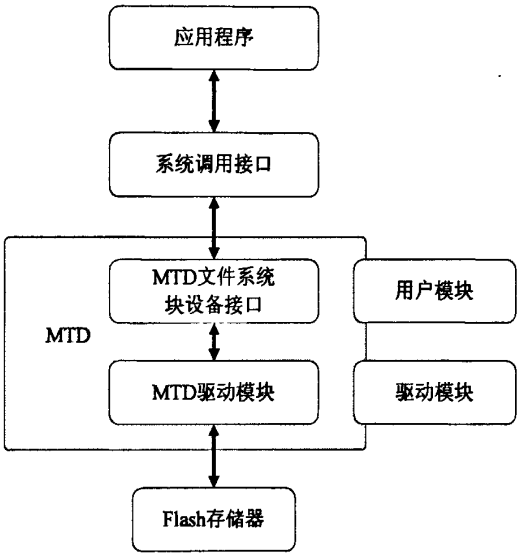

图 3-3 MTD 与 Flash 和文件系统的关系

Fig3-3.Relationship between Flash and FileSystem through MTD

平台采用的 JFFS2[17]（Journalling Flash FileSystem2）文件系统，即日志闪存文件系统，是基于 NAND Flash 的文件系统。日志文件系统的主要思想是通过在存储系统利用一系列的节点记录对文件进行操作，记录文件系统的变化。JFFS 文件系统是最早由瑞典 Axis Communications 公司开发的基于 Linux2.0 的嵌入式文件系统。JFFS2 是 RedHat 公司在 JFFS 基础上开发的，针对 RedHat 公司的嵌入式产品 eCos 的文件系统。JFFS2 的特点是：

- 可读写、支持写平衡；
- 支持数据压缩；
- 采用非顺序的基于哈希表的日志节点结构，节点的操作速度加快；
- 提供掉电安全保护机制；
- 支持多种节点类型；
- 提高闪存利用率，从而降低了内存的损耗。

以上特点决定了 JFFS2 完全符合嵌入式系统对文件系统的要求，同时还方便了程序员的开发，因此选用 JFFS2 作为本平台的文件系统。

3.2.2 系统软件平台的移植

ARM 节点软件系统平台的移植步骤如下：

1、移植 Bootloader

（1）选用开发板自带的 BIOS 源代码，根据实际需要作出相应的修改，首先配置内核启动参数，Nand.c 文件中的 LoadRun 函数：

```
Char *linux_params="root=/dev/mtdblock2 rootfstype= jffs2
console=ttySAC0，115200 init=/linuxrc mem=64M"
```

（2）根据 S3C2440 的数据手册修改时钟参数，在文件 2410bios.c 中修改如下：

```
ChangeMPLLValue（120，2，1）;
SetClockDevider（3，1）;
```

（3）编译 BIOS 源码。在 CodeWarrior（ARM 集成开发调试工具）中创建新项目，相关编译参数选择为 ARM920T，然后编译，生成 bin 文件。

（4）通过 JTAG 调试器和 ARM 节点上的 JTAG 接口，将 bin 文件烧写到 ARM 节点的 Nand Flash 中，烧写的起始地址为 0x0000000。

Boot Loader 正常运行后的界面如图 3-4 所示。

```
************************************
*     BJTU NGIRC WSN               *
*     GEC2440 Board Loader V1.1    *
************************************

Power on reset
Read chip id = ec76
Nand flash status = c0, NandAddr=1
Env.Os_Auto_Flag=1
NAND Flash Boot
Read Norflash ID is : 0x225b
AM29LV800BB found at nGCS2

Please select function :
0 : USB download file
1 : Uart download file
2 : Write Nand flash with download file
3 : Load Pragram from Nand flash and run
4 : Erase Nand flash regions
5 : Write NOR flash with download file
6 : Set boot params
7 : Set AutoBoot parameter,1:linux 2:wince
```

图 3-4 Boot Loader 运行界面

Fig3-4.Interface of Running Boot Loader

Boot Loade 还提供一些便捷的功能：USB 下载、串口下载、Nand Flash 写操作、Nand Flash 擦除、Nor Flash 写操作、启动参数设置和自启动选项。之后的 Linux 内核和文件系统的移植都是基于功能“0：USB download file”和“2：Write Nand flash with download file”的。

2、移植 Linux 内核

（1）下载内核源码 linux-2.6.22.19.tar.gz（2.6.17 之后的内核版本都可以），解压后进入内核目录。

（2）修改内核源码目录下的 Makefile 文件，指定交叉编译的类型和路径：

ARCH = arm

CROSS_COMPILE = /path-of-the-cross-compiler/arm-linux-

（3）修改内核代码中的 arch/arm/plat-s3c24xx/common-smdk.c 文件，根据 BootLoader 中 Nand Flash 分区修改文件中的 Nand Flash 的分区信息和 Nand Flash 的硬件信息，具体分区信息如表 3-1 所示：

表 3-1 Nand Flash 分区

Tab3-1 Division of Nand Flash

名称	起始地址	分区大小	offset 参数的配置	Size 参数的配置
Boot Loader	0x0000000	192K	0	SZ_128K+ SZ_64K
Linux 内核	0x0030000	1856K	SZ_128K+ SZ_64K	SZ_1M*2- (SZ_128K+ SZ_64K)
根文件系统	0x0200000	60M	SZ_1M*2	SZ_1M*60

（4）修改 drivers/mtd/nand/s3c2410.c，禁止内核的 ECC 校验：

chip->ecc mode=NAND_ECC_NONE

（5）拷贝 s3c2410 开发板的默认配置到内核根目录下，以简化配置过程：

#cp arch/arm/configs/s3c2410_defconfig .config

（6）内核配置 make menuconfig：根据开发板的特点与所需的功能配置相关的 System Type，File System，Device Drivers 中的选项等，相关选项可以选择为编译进内核<*>，也可以选择为编译成模块形式<M>，内核配置页面如图 3-5 所示。

（7）修改 Boot options：制定文件系统分区、串口显示设备、文件系统启动文件以及内存大小。

root=/dev/mtdblock2 console=ttySAC0，115200 init=/linuxrc mem=64M

（8）保存后退出，在/arch/arm/boot 中生成内核映像文件 zImage：

#make zImage

（9）通过 Boot Loader 的 USB 下载功能将内核映像文件下载到 SDRAM，然后再通过烧写功能将内核映像文件搬移烧写到 Nand flash 中的第二个分区，至此已经成功移植 Linux 内核。

3、移植 JFFS2 文件系统

（1）创建根文件系统目录和设备节点，具体操作如下：

#mkdir /rootfs

#cd /rootfs

#mkdir bin sbin lib home tmp usr mnt var

#cd dev

#mknod console c 5 1

#mknod null c 1 3

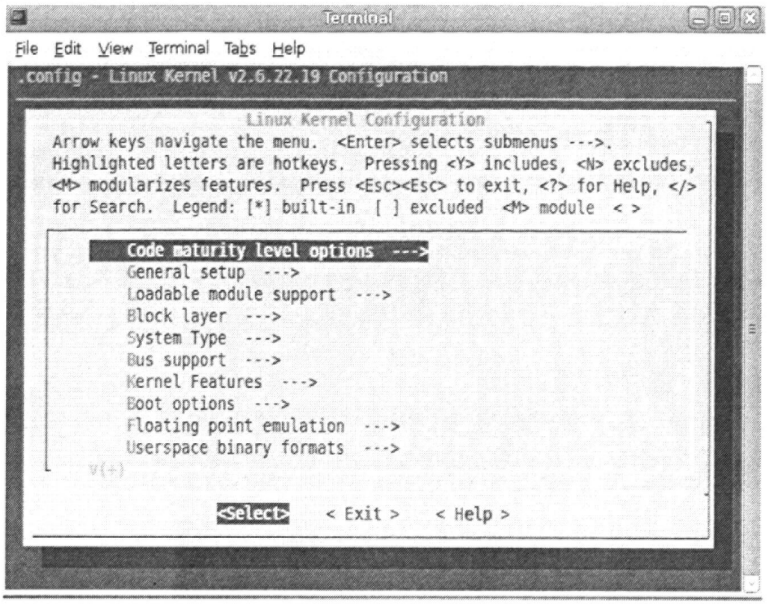

图 3-5 内核配置界面

Fig3-5.Configure Interface of Kernel

（2）准备库文件及 GNU 常用工具和 Shell 脚本工具，BusyBox 是一个 Unix 工具集，并且提供了类似于 Linux 内核的配置菜单，可以方便选择所需工具和库文件。交叉编译后生成相应的 Linux 常用工具及库文件，将这些复制到上述步骤创建的系统目录的对应文件夹下。

（3）编写文件系统启动需要的文件：/inuxrc、/etc/init.d/rcS、/etc/fstab、/etc/inittab。/inuxrc 指明将要挂载的文件系统并启动系统第一个 init 进程。在 rcS 文件中可以添加用户想要执行的操作，比如用户可以要求在系统启动时自动加载某个驱动。

（4）使用 mkjffs2 工具生成文件系统映像文件 root_jffs2.img。

#./mkcramfs /rootfs cramfs.img

（5）下载文件系统映像文件到开发板，烧写到 Nand flash 的第三个分区。

具备了 Boot Loader，Linux 内核和 JFFS2 文件系统后，开发板的系统便可以正常启动，为后续的驱动加载奠定了软件平台基础。

3.3 初始化模块

初始化模块包括 IEEE 802.15.4 的 MAC 层初始化和 IEEE802.11g 无线网卡的

初始化。此外还为其他模块提供了套接字、内核定时器等初始化功能，将在后面章节的相应的小节进行详细介绍。

3.3.1 IEEE802.15.4 的 MAC 层初始化

IEEE 802.15.4 的 MAC 层初始化主要是完成硬件的初始化和任务调度初始化的工作。初始化的过程包括以下步骤：

- I/O 寄存器的初始化

I/O 寄存器的初始化主要是对涉及到的每一个寄存器基于内存映射方式的映射。其中自定义的函数 regspi_ioremap()、regtimer_ioremap()和 regint_ioremap()都是利用 Linux 提供的 ioremap()函数，分别对 SPI、时钟中断管理和相关管脚的 IO 地址空间进行映射。

void * __ioremap(unsigned long phys_addr，unsigned long size，unsigned long flags)

该函数负责把一个物理内存地址映射为一个内核的虚拟地址空间，即一个内核指针，phys_addr 为映射的起始地址，size 为要映射的空间大小。

- 端口的初始化

Port_Config() 函数负责初始化 S3C2440 通用 I/O 口，包括端口引脚功能设置置等。

- 中断的初始化

利用内核提供的 set_irq_type()和 request_irq()内核函数，向内核申请中断信号线，并设置相应的外部中断。

- 时钟的初始化

利用内核提供的 clk_get ()、clk_enable ()等内核函数，为 SPI 向内核申请、使能系统时钟。

- MAC 层状态的初始化

MAC_INIT()负责完成 MAC 时钟、MAC 层各任务状态位的初始化。

- SPI 初始化

SPI_INIT()负责设置 SPI 传输波特率、SPI 传输时序等参数，以完成 SPI 的初始化功能。

- CC2420 芯片的驱动

CC2420_PowerUp() 函数首先 CC2420 芯片晶振起振，并通过状态字节的读取、判断 CC2420 晶振状态位，确保 CC2420 芯片晶振起振后，CC2420 芯片开始工作，改变 MAC 相关的状态位，记录 CC2420 的工作状态。

- 任务调度的初始化

通过内核定时器的申请和初始化，实现对 IEEE 802.15.4 MAC 层协议各任务进行调度。

➢ IEEE802.15.4 设备驱动的初始化

检测设备是否存在，并向内核注册字符型设备驱动，分配内存空间，检测设备 I/O 端口和中断号。

3.3.2 IEEE802.11g 无线网卡初始化

IEEE802.11g 无线网卡的初始化过程包括无线网卡驱动的安装以及配置。

➢ IEEE802.11g 无线网卡驱动的安装

IEEE802.11g 无线网卡驱动的安装过程主要包括无线网卡驱动的移植和内核配置。

1、解压驱动源代码并拷贝至内核文件目录下：

```
#cp 2009_02066_RT73_linux_STA_1.1.0.2/Module  ../linux2.6.24/drivers/net/wirless/rt73
```

2、修改驱动的Makefile：

```
#cp Makefile.6 ./Makefile
```

Makefile 的修改主要是指交叉编译，以及编译内核的目录修改，修改内容如下：

```
CC    :=arm-linux-gcc
LD    :=arm-linux-ld
#PLATFORM=PC
PLATFORM=CMPC
.....
ifeq ($(PLATFORM), CMPC)
LINUX_SRC = /home/cuijie/linux-2.6.22.19
.....
```

3. 修改内核/usr/src/linux-source-2.6.24/drivers/net/wireless/目录下的 kconfig 和 Makefile：

修改 Kconfig，以便在内核配置中增加 RT73 网卡驱动的选项，在 Kconfig 的末行添加：

```
config RT73
        tristate "support rt73 wireless usb network device"
        depends on USB && NET && USB_USBNET
```

修改 Makefile，在内核编译的同时也可以编译无线网卡的驱动，在 Makefile 末尾处添加：

obj-$(CONFIG_RT73)　　+= rt73/

4．配置内核，添加无线网卡及其依赖的内核选项：

选上如下选项：

<*>sungsang24xx——选定 CPU 芯片类型

<*>S3C2410 DMA support——选定内核支持动态内存存取

<*>SMDK2440——选定硬件平台

<*> multi-purpose USB Networking Framework——选定内核支持 USB 类型的网络设备

<*> network console logging support——netpoll 让内核在网络和 I/O 子系统尚不能完整可用时，依然能发送和接收数据包

<M> support rt73 wireless usb network device——选定 RT73 无线网卡以驱动的形式编译

5．重新编译内核，编译驱动模块：

#make zImage

#make modules

将生成的内核镜像文件烧写进 S3C2440 的 NAND Flash 中，并将生成的无线网卡驱动文件 rt73.ko 下载到目录/home 下，手动 insmod 加载驱动，完成 IEEE802.11g 无线网卡驱动的安装。

#insmod rt73.ko

➢ 无线网卡的配置

利用无线网络配置工具对 IEEE802.11g 网络进行配置。

1、wireless-tools的移植

由于S3C2440的文件系统较为简洁，因此无线网络的配置工具通常需要另外的移植。

下载无线网络工具压缩包wireless_tools.29.tar.gz并解压，将编译工具改为交叉编译链后再进行编译，在目录下产生iwconfig、iwevent、iwgetid、iwlist、iwspy、iwpriv等可运行文件，直接拷贝到S3C2440的/sbin目录下即可。此外把/lib下的libiw.so.29库文件拷贝到S3C2440的/lib下。

2、无线网卡的配置

#ifconfig rausb0 up——启动无线网卡，其设备名称为 rausb0

#iwconfig rausb0 add 3ffe:3240:8007:1005::2000——根据节点的接口标识符配置IEEE802.11g 无线网络地址

#iwconfig rausb0 essid wsn mode ad-hoc——设置节点无线网卡的通信模式为 ad-hoc，接入网络名称为 wsn

为方便使用，无线网卡的配置步骤可以写进脚本，每次只需运行脚本就可以完成配置。正确配置后的无线网卡状态如图 3-6 所示。

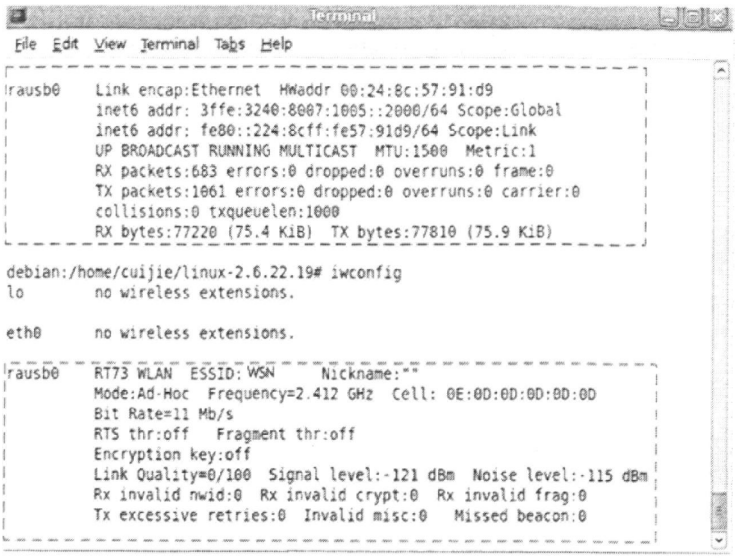

图 3-6 无线网卡状态

Fig3-6.Status of Wireless Network Card

3.4 任务调度模块

异构无线传感器网络多链路传输系统涉及到多个功能模块和繁杂的任务，因此需要任务调度模块对各个功能模块协调使用，以保证每个功能模块和任务都能高效有序地运作。

引入内核线程和内核定时器，利用 Linux 内核的调度机制，对提高软件运行整体的效率和条理性都起到了巨大的作用，也为其他功能模块提供了更加简洁的调度功能。

3.4.1 内核线程的创建和使用

内核线程是由内核直接控制的，可以看作是内核的分身，一个分身用来处理一件任务。内核线程的引入为处理异步事件提供了更加方便的使用。内核线程具有以下性质：

- 通过系统调用函数 clone()来实现的，与调用它的进程具有相同的进程空间；
- 在内核空间内运行，所以内核线程可以访问内核中数据，直接调用内核函数，并且在运行过程中不能被抢占。
- 具有自己的上下文，类似于用户进程，同样被进程调度程序调度，也可以睡眠，不同之处就在于内核线程运行于内核空间；

- 只可以访问虚拟地址空间的内核部分（高于 TASK_SIZE 的所有地址），因而没有独立的地址空间。

Linux 内核中用 task_struct 数据结构来表示一个内核线程。通过这个结构，内核就可以实现对内核线程的操作和控制。task_struct 数据结构可以唯一标识一个内核线程。在这个结构中包含了内核线程相关的所有信息，包括线程状态、调度信息、处理器信息、线程的归属所有信息、时间和定时器、内存管理信息、信号处理信息、文件系统信息等。

调用 kernel_thread 函数可建立一个内核线程，同时会为这个内核线程申请一个 task_struct 数据结构，函数原型如下：

<asm-arch/processor.h>

int kernel_thread(int (*fn)(void *), void * arg, unsigned long flags)

指针 fn 指向了要执行的函数，arg 代表传递给该函数的参数，标志位 flags 可以定义为：

CLONE_VM 0x00000100 ——内核线程与父进程共享内存

CLONE_FS 0x00000200 ——内核线程与父进程共享文件描述符

CLONE_FILES 0x00000400——共享文件

CLONE_SIGHAND 0x00000800——共享信号

CLONE_PID 0x00001000 ——共享 PID

CLONE_VFORK 0x00004000——父进程在内核线程释放内存前不做任何动作

CLONE_THREAD 0x00010000 ——是否是同一线程组

本论文在主线程中完成数据的定时发送功能，同时由内核子线程负责循环检测任务队列。内核线程的使用如图 3-7 所示。

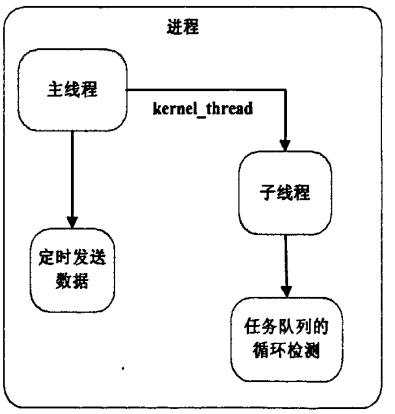

图 3-7 内核线程

Fig3-7. Kernel Thread

3.4.2 内核定时器的创建和使用

内核定时器用来在未来的某个特定时间点调度执行某个函数，从而可用于完成许多任务。每个内核定时器都对应一个 timer_lis 数据结构，存放定时器的名称、执行函数、超时时间、传递参数等。执行函数是异步运行的，不会与当前父进程同时运行，类似于中断的产生。

本论文利用内核定时器来实现任务队列周期轮询，并实现了数据的定期发送。下面以内核定时器来实现数据定期发送为例介绍它的创建和使用。

每个定时器都对应一个 timer_list 结构体，用于存放内核定时器相关的各种信息，因此先要定义 timer_list 结构体。

```
static struct timer_list aROUNDTIME;
```

调用内核函数 init_timer( )对 aROUNDTIME 进行初始化,分配内存空间。

```
init_timer(&aROUNDTIME);
```

指定内核定时器的各项参数。aROUNDTIME.data 是指定的要传递的参数，赋值为 0。参数 aROUNDTIME.function 指向定时器超时时要执行的函数，这里为 aROUNDTIME_func 函数，主要功能是检查发送或接受队列是否为空，若不为空，则进行下一步的处理。

参数 aROUNDTIME.expires 表示期望定时器执行的 jiffies 值，到达 jiffies 值时，将调用 aROUNDTIME_func 函数，并传递 aROUNDTIME.data 作为参数。内核定时器的时间是以 Linux 内核的时钟滴答为单位的。在内核空间中，jiffies 是 Linux 内核中的一个全局变量，用来记录自系统启动以来产生的节拍的总数。jiffiers 由变量 CONFIG_HZ 的倒数得来。而在 ARM 平台的 Linux 中定义 CONFIG_HZ=200，因此可以计算出时钟滴答数为 5 毫秒，也就是说内核定时器一个单位为 5 毫秒。启动时，内核将该变量初始化为 0，此后，每次时钟中断处理程序都会增加该变量的值。因此 jiffies + 20 表示推后 100 毫秒。

```
aROUNDTIME.function=aROUNDTIME_func;
aROUNDTIME.expires=jiffies+20;
```

内核函数 add_timer()将定时器添加到定时器等待队列中，开始计时。100 毫秒到，执行超时函数。

```
add_timer(&aROUNDTIME);
```

内核定时器本身并不是周期运行，它在超时后会自动销毁。因此，如果要实现周期轮询，就需要在定时器执行函数返回前再次激活定时器。也就是说，在退出 aROUNDTIME_func 函数之前，要执行以下两行代码：

```
aROUNDTIME.expires=jiffies+20;
```

```
add_timer(&aROUNDTIME);
```

重新执行超时时间值，再次启动内核定时器，按新的超时值开始重新计时。可以根据具体实践的需要，设置 expires 值，实现不同时间周期的周期轮询。

3.5 数据传输模块

数据传输模块作为多链路传输技术中最重要的模块之一，完成了网络层数据的发送和接收功能。具体而言，在接收数据时，在适配层对数据进行了解封装，提取出其中的 IPv6 数据包，交给 IPv6 协议栈处理。发送数据时，应用层的数据需要经过统一的封装，采用内核套接字完成内核在协议栈中的进一步封装和处理。这两个过程中，都需要有 Netfilter 的协助，以完成数据包在不同层次间的传递。

3.5.1 发送子模块的设计和实现

基于内核协议栈的网络层空间的发送子模块是采用内核空间的 UDP 套接字实现的，简单型传感数据采集完成后，数据就会送往发送子模块进行处理。

发送子模块可以分成两部分：内核 UDP 套接字数据发送和适配层的发送处理。内核 UDP 套接字发送主要负责创建内核 UDP 套接字，并按照统一的格式对数据进行应用层的封装，然后将数据发送出去，即交给内核协议栈处理。适配层的发送处理指的是数据在进入路由前，由钩子函数取出，交给适配层处理，适配层将根据数据包的类型，完成相应的工作。

3.5.1.1 内核 UDP 套接字数据发送的实现

在 Linux 中，用户空间的 SOCKET 编程，数据发送过程中，发送数据将被整理为 msghdr 的数据结构，然后调用 sock_sendmsg()，这个函数完成的工作包含两部分，一是将整理好的 msghdr 数据结构从应用层传递到 inet 层；二是根据 msghdr 数据结构中的数据包创建 sk_buff 数据结构，填充相应字段，sk_buff 最终在 Linux 内核协议栈中一层一层向下传递封装。msghdr 结构可以看作是数据在应用层的封装。msghdr 的定义如下：

```
struct msghdr {
    void    *msg_name; ——指向目的地址结构的指针
    int     msg_namelen; ——地址结构的长度，对于 IPv6 地址，长度为 16
    struct iovec    *msg_iov; ——指向要发送的数据缓存
```

```
__kernel_size_t msg_iovlen; ——数据长度
void    *msg_control; ——控制信息
__kernel_size_t msg_controllen; ——控制信息长度
unsigned   msg_flags; ——标志位，MSG_DONTWAIT，表示使用非阻塞操作
};
```

在内核中，还定义了 kvec 数据结构，用于记录消息的内容和长度，定义如下：

```
struct kvec{
    void __user    *iov_base;
    __kernel_size_t iov_len;
};
```

发送子模块是在内核空间中进行的。因此对于采集到的简单型传感数据，在内核空间直接构造 msghdr 数据结构，msghdr 中有专门的数据指针，指向要发送的数据，然后填充其中结构中的各参数变量，而后创建 UDP 类型套接字，将 msghdr 发送出去。不同于用户空间的套接字使用，在内核空间有自身的一系列套接字的操作函数，需要设置正确的套接字参数：端口、目的地址和协议类型，完成套接字的创建和发送过程。由于 ARM 节点具有两个 MAC，因此也具有两个不同的网络地址，根据多链路选择的结果绑定不同的网络地址。

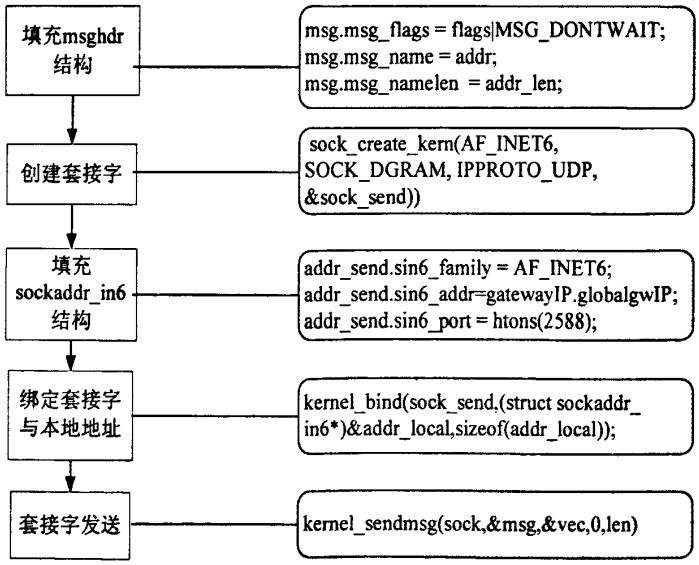

图 3-11 UDP 套接字的发送过程

Fig3-11. Send Procedure of UDP Socket

数据进入内核协议栈之后，内核会根据参数定义，对数据包进行传输层和网络层的封装，添加 UDP 头部和 IPv6 头部。这个过程对于用户来说是完全透明的。通过钩子点的设置，也就是数据包处理过程，可以将封装完成的数据包取出，若

该数据包是在 IEEE802.15.4 链路上传输，则将数据包添加到适配层的发送队列中，交给 IEEE802.15.4 的适配层进一步处理；否则，数据包返回内核协议栈，直到发送过程的完成。内核 UDP 套接字的发送过程如图 3-11 所示。

3.5.1.2 适配层的发送处理

适配层是网络层与 IEEE802.15.4MAC 层之间的桥梁，对于不同的数据包类型，适配层将分别进行处理。

适配层的发送过程由内核定时器进行超时控制。超时后会执行 MARP 路由更新程序，否则就检查发送队列是否为空。适配层发送处理流程如图 3-12 所示。具体过程如下：

1、判断是否超时：
   - 若超时，则进入 MSRP 路由更新处理，同时定时器置 0，开始重新计时；
   - 若未超时，则检查发送队列；

2、判断发送队列是否为空：
   - 若发送队列为空，则返回；
   - 若不为空，检查数据的跳数字段；

3、判断跳数是否为 0：
   - 若为 0，则丢弃该数据包；
   - 若不为 0，则为数据包添加适配层头部，并检查数据包类型，判断是路由包还是数据包；

4、判断数据包的类型：
   - 若为路由包，则对路由包设置其目的地址即可，且不含最终目的地址域，然后返回继续进行处理；
   - 若为数据包，将查找路由表项；

5、判断是否存在路由：
   - 若不存在到达最终目的地址的路由，则先将数据包缓存后，发送路由 RREQ 请求；若为数据包，将查找路由表项；
   - 若存在路由，会进一步判断跳数；

6、判断跳数是否为 1：
   - 如果跳数为 1，说明数据包经过一跳即可以到达 Sink 节点，无需包含最终目的地址域；
   - 如果跳数超过 1，则在构造适配层数据包过程中，应当包含最终目的

地址域。

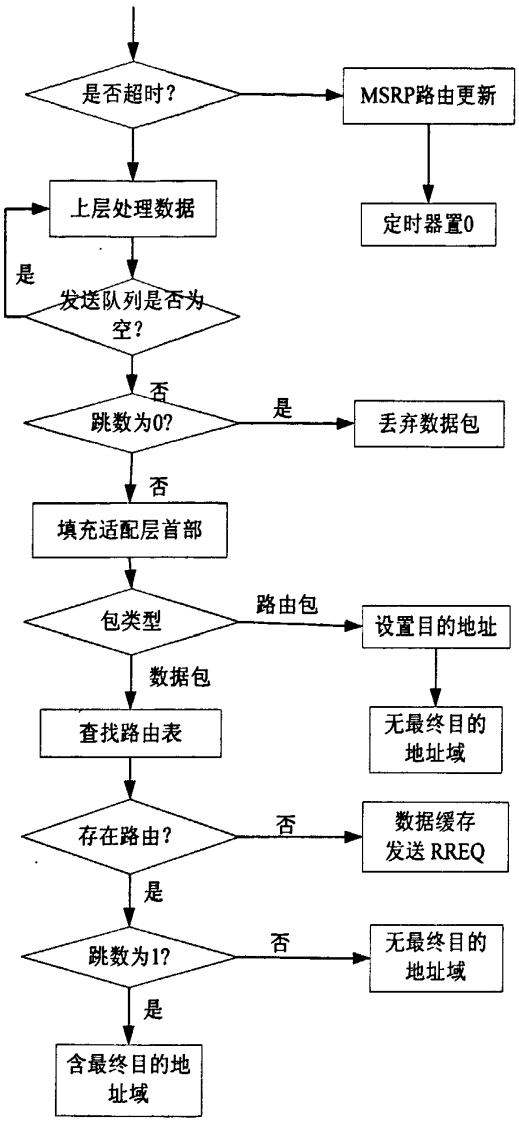

图 3-12 适配层发送处理流程

Fig3-12. Send Procedure of Adapt Layer

3.5.2 接收子模块的设计和实现

数据传输模块的接收子模块负责完成各种数据的接收过程。利用 Linux 的 sk_buff 机制设计并实现接收子模块。

3.5.2.1 接收子模块设计

数据的接收由 IEEE802.15.4MAC 层的中断控制程序控制。对于需要本地处理

的数据包，这里用到了 Linux 网络协议栈通用的数据结构 sk_buff。sk_buff 是 Linux 网络协议栈中一个重要的数据结构，用于存储数据包的各类信息，包括每层网络协议的头部、负载、协议类型、地址信息等。协议栈中的各层协议都可以通过对 sk_buff 的操作实现本层协议数据的添加或提取，数据包在 Linux 协议栈中的不同协议层之间并不会来回的复制，而是修改 sk_buff 中的 data、len、tail 等指针，就可以实现对各层协议对数据包的处理，这种机制可以避免系统资源和空间的浪费，同时还提高了执行效率。内核在各个协议层还提供了一系列的函数对 sk_buff 进行操作。

利用 sk_buff，首先向内核申请一个该数据结构：

```
skb=dev_alloc_skb(len);
```

然后根据协议类型等填充 skb 数据结构的相应字段，并将数据拷贝至该结构的数据部分：

```
memcpy(skb_put(skb,len),buf,len);——将数据拷贝到 skb 中
struct ipv6hdr *ip6h = (struct ipv6hdr *)skb->network_header;
skb->protocol = htons(ETH_P_IPV6);——协议类型为 IPv6
skb->pkt_type = PACKET_HOST;——由本地主机处理的数据包
memset(skb->cb, 0, sizeof(struct inet6_skb_parm));
skb->dev = dev_get_by_name("eth0");——虚拟为以太网设备传来的数据
skb->ip_summed = CHECKSUM_UNNECESSARY;——ip 校验
if (skb->dst) dst_release(skb->dst);
skb->dst = NULL;
```

3.5.2.2 接收子模块实现

接收子模块将根据包的类型对包采取不同的处理。接收子模块的处理流程如图 3-13 所示。具体过程如下：

1、判断数据包类型：

- 若为路由包，则将交给 MARP 路由完成处理；；
- 若为数据包，则查找路由表；

2、判断数据包是否含最终目的地址域：

- 若无最终目的地址域，则填充 skb_buff 数据结构，交给内核协议栈处理
- 若有最终目的地址域，则判断数据发送队列是否为空，进入数据发送过程；

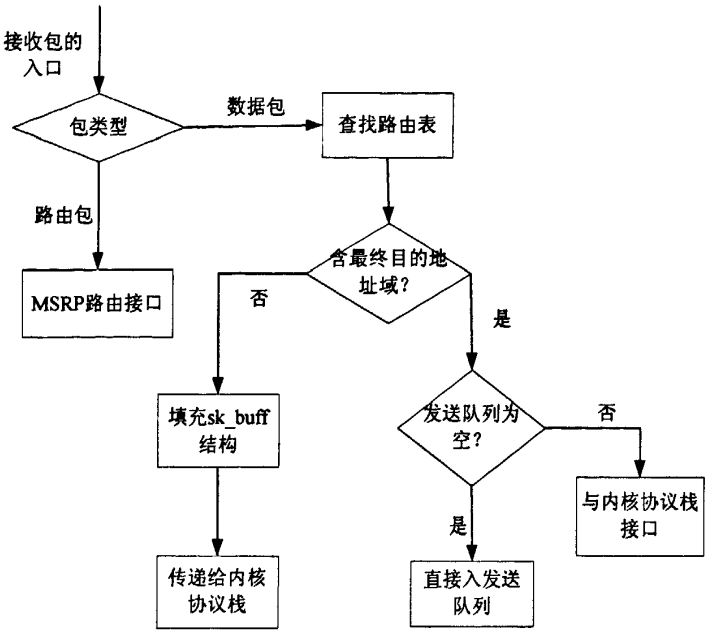

图 3-13 接收子模块处理流程

Fig3-13. Recieve Procedure of Sub-Module

3.5.3 数据包处理的设计与实现

3.5.3.1 数据包处理机制

数据传输模块中的数据包的处理利用的是 Netfilter。Netfilter[28]是 Linux 内核中的一个包过滤框架，为用户提供一个不同于 BSD Socket 接口的操作网络数据包的机制。在这个框架上可以进行包过滤、状态检测、网络地址转换和包标记等多种功能。Netfilter 为协议栈中的每种协议都定义了若干个钩子（HOOK），如 IPv6 就定义了五个钩子，相应协议的数据包在网络层次之间的传递过程中会通过若干个钩子，每一个钩子可以设置一个用户自定义的处理函数。内核模块通过在各个钩子上注册处理函数，以对经过该钩子的数据包进行操作。处理完成后，根据内核提供的选项返回给内核进行下一步的处理。

IPv6 协议的 Netfilter 结构如图 3-8 所示：IPv6 的 5 个钩子函数，分别为：

- NF_IP_PRE_ROUTING：接收到的数据包在进入路由选择处理之前经过这个钩子；
- NF_IP_LOCAL_IN：对于需要本地处理的数据包，即本节点为目的节点，可以在 NF_IP_LOCAL_IN 钩子处截取数据包；
- NF_IP_FORWARD：接收到的数据包如果需要转发，将经过该钩子点；

- NF_IP_POST_ROUTING：本节点发送或转发的数据包在离开之前，都会进过这个钩子；
- NF_IP_LOCAL_OUT：由本地上层协议封装后要发送的数据包在路由选择之前，进过该钩子，因此可以利用该钩子进行网络地址转换等操作。

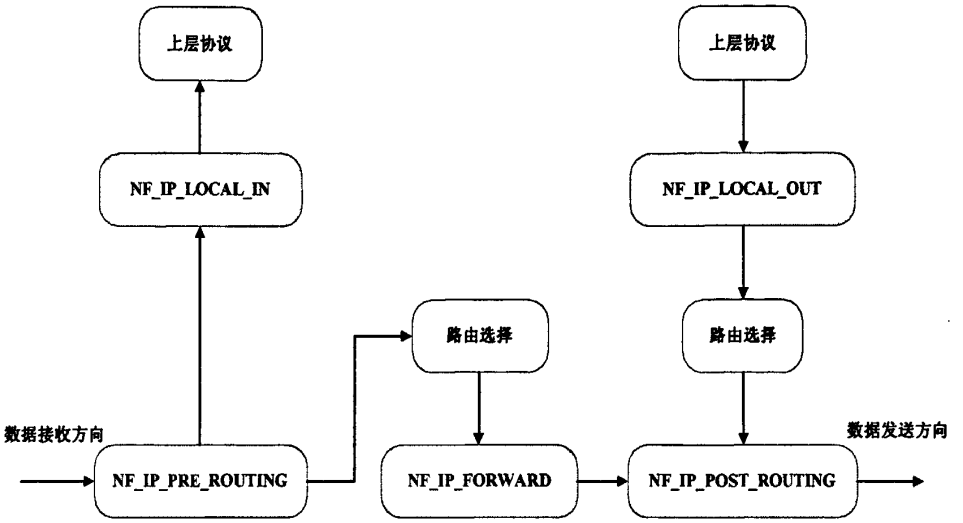

图 3-8 IPv6 协议的 Netfilter 结构

Fig3-8. Structure of IPv6 Netfilter

内核模块可以注册挂接一个或多个这样的钩子函数，在数据包经过这些钩子点时，调用相应的处理函数被，Linux 内核还定义了 Netfilter 的返回值：

- NF_ACCEPT：将数据包返回内核协议栈，继续下一步的处理；
- NF_DROP：丢弃该数据包，不再进行任何处理；
- NF_STOLEN：表示内核模块会保留数据包，不再返回内核协议栈，或者发生了异常分组的问题；
- NF_QUEUE：进入用户空间的数据包队列，可以在用户空间对数据包进行操作；
- NF_REPEAT：重新调用钩子函数再次处理；

Netfilter 的钩子同样具有一个特定的数据结构 nf_hook_ops，用于表示钩子的各种信息。.hook 指向用户自定义的钩子函数，.pf 表示协议类型，.hooknum 则表示钩子的类型，.priority 表示在运行时这个钩子函数执行的优先级顺序。nf_hook_ops 数据结构的参数定义如下：

```
static struct nf_hook_ops sensordata_out =
{
    .hook     =    kernel_process_out,
    .pf       =    PF_INET6,
```

```
    .hooknum     =    NF_IP_LOCAL_OUT,
    .priority    =    NF_IP_PRI_FIRST,
};
```

钩子函数也具有固定的定义格式，kernel_process_out 的定义如下：

```
static unsigned int kernel_process_out(unsigned int hook, struct sk_buff **pskb,
                                       const struct net_device *indev,
                                       const struct net_device *outdev,
                                       int (*okfn)(struct sk_buff *))
```

3.5.3.2 数据包处理的实现

本论文中，利用钩子主要实现了数据包的截取，以及网络地址转换功能。在数据接收方向，设置 NF_IP_PRE_ROUNTING 类型的钩子点，主要是针对服务器通过网关发来的命令数据信息。此处的钩子函数首先判断数据包是否是由服务器端发送的命令信息，如果是，则提取出信息的 IPv6 数据包，交给服务器命令处理函数进行进一步处理，同时丢弃原数据包；否则，将该包返回内核协议栈继续处理。NF_IP_PRE_ROUNTING 钩子处理过程如图 3-9 所示。

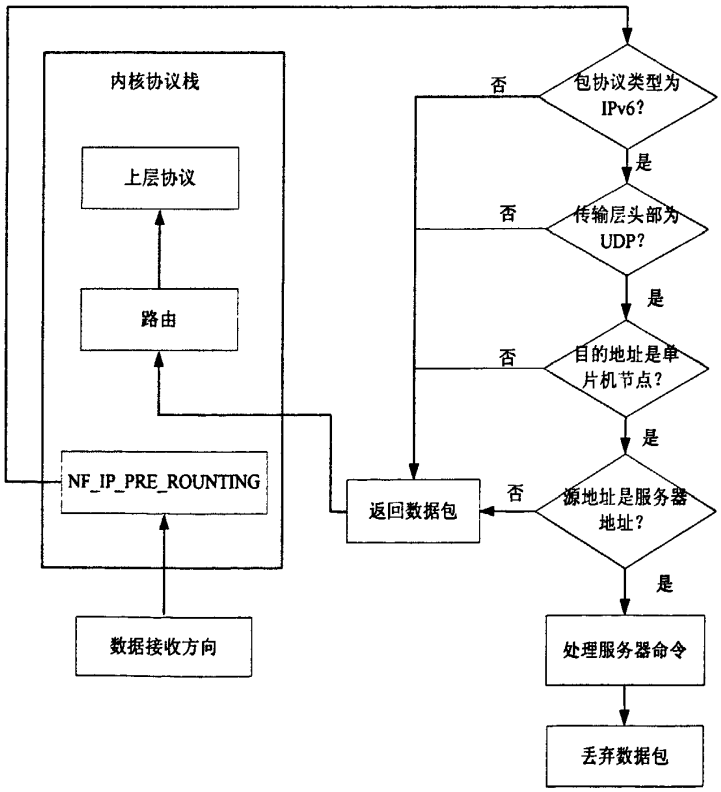

图 3-9 NF_IP_PRE_ROUNTING 钩子处理过程

Fig3-9. Process Procedure of NF_IP_PRE_ROUNTING Hook

数据的发送方向设置了 NF_IP_LOCAL_OUT 类型钩子，在数据包进入路由处理前截取数据包，钩子函数检查其中的切换标志位，若该数据包需要由 IEEE802.15.4 协议切换到 IEEE802.11g，则钩子函数根据地址转换规则，替换数据的源地址和目的地址，然后数据包返回，在内核中继续向下层协议传递。NF_IP_LOCAL_OUT 钩子处理过程如图 3-10 所示。

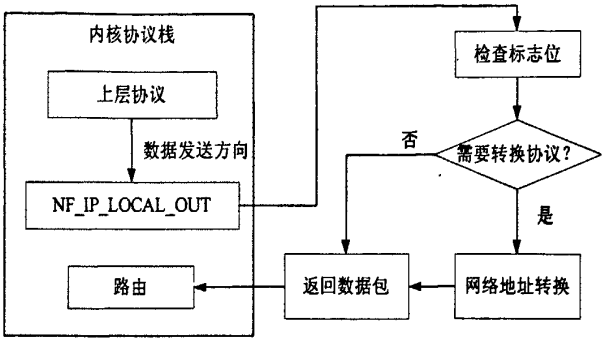

图 3-10 NF_IP_LOCAL_OUT 钩子处理过程

Fig3-10. Process Procedure of NF_IP_LOCAL_OUT Hook

3.5.4 数据传输模块的重要数据结构设计

数据传输模块中的适配层相关的处理部分涉及到很多重要数据结构，下面对这些数据结构做简要的介绍。

1、IEEE802.15.4 的适配层头部数据结构

```
struct adapt_unfrag_header{
    u8_t LF:2;
    u8_t prot_type1:6;
    u8_t prot_type2:6;
    u8_t M:1;
    u8_t rsv:5;
    struct Mesh_delivery_field MDField;
  }unfrag_header;
```

2、发送队列中包的数据结构

```
typedef struct tx_entry{
     ADDRESS txq_DestiAddr;——数据包目的地址
     BYTE toport;——目的端口
     u8_t   txq_hops;——跳数
     u8_t   txq_addr_mode;——地址类型，包括短地址和长地址
```

```
    u8_t   txq_pkt_type;——包类型，包括路由包、数据包等
    u8_t   txq_data_len;——包长度
    u8_t   txq_data[TXBUFLEN];——数据部分
  }tx_queue;
```

3、发送队列的数据结构

```
struct tx_queue{
  tx_entry txbuf[TXBUFNUM];
  u8_t flags;——队列初始化时置 0，用于记录发送队列中包的排队
  };
```

4、接收队列中包的数据结构

```
typedef struct rx_entry{
      BYTE   fromPort;——源端口
      ADDRESS aTargetMacaddr;——目的地址
      UINT8 minLQI;
      u8_t flags;——收标志位，若加入接收队列，则置 1
      u8_t buflen;——包长度
      u8_t pdatabuffer[RXBUFLEN];——包内容
  }rx_entry;
```

5、接收队列的数据结构

```
struct queue_buffer {
   rx_entry rxbuf[RXBUFNUM];
   u8_t flags;——队列初始化时置 0
 };
```

3.6 数据采集模块

采集简单类型数据的传感器模块可以直接通过 PF 口连接到微处理芯片上，完成数据的传感和采集。对于 ARM 节点等高端节点，多媒体数据的采集成为了它的优势。因此本小节主要介绍了图像数据采集的设计与实现。首先介绍 Linux 中视频图像处理的 API 接口 V4L 技术。在 V4L 的基础上，进一步介绍了图像数据采集的实现流程。

3.6.1 视频设备接口设计

视频设备接口采用 V4L。V4L[29]，其全称是 Video4Linux（Video for Linux），是在 Linux 内核中关于视频设备的 API 接口，是 Linux 控制摄像头等设备的基础。在程序中调用 V4L 的提供功能函数即可实现对摄像头设备的操作。V4L 为用户提供了通用的图像数据结构，并且还有大量的可调用功能函数，此外，可以设置图像的大小、通道、帧格式等一系列相关信息。因此，用户可以方便地根据自己需求，实现图像的抓取、控制等功能。V4L 获取图像的方式有两种：mmap（内存映射方式）和 read（直接读取方式）：直接读取方式通过内核缓冲区来读取图像数据；内存映射方式无需读取缓存，而是将文件直接映射到内存中，利用指针读取，读取速度更快。

V4L 提供的通用视频设备数据结构 v4ldevice，可以管理视频设备的相关信息，v4ldevice 数据结构定义如下：

```
typedef struct _v4ldevice{
    int fd; --设备号
    struct video_capability capability;——设备的基本信息，包括名称、分辨率、信号源等
    struct video_channel channel[10];——信号源属性，包括信号源编号、名称、类型等
    struct video_picture picture;——设备采集的图像属性，包括亮度、对比度、格式、色度
    struct video_clip clip;
    struct video_window window;——图像捕获域信息，包括捕获图像的高和宽等
    struct video_capture capture;
    struct video_buffer buffer;——用于帧的存储
    struct video_mmap mmap;——mmap 方式获取数据时，存储在内存中的帧信息
    struct video_mbuf mbuf;——该结构包括内存映射区域的大小、捕捉设备缓存帧的数目、
                              偏移地址
    struct video_unit unit;
    unsigned char *map;——mmap 方式获取数据时，数据的首地址
    pthread_mutex_t mutex;
    int frame;
    int framestat[2];
    int overlay;
}v4ldevice;
```

在内核配置时，选择 V4L 相关选项，保证内核对 V4L 的支持，V4L 的内核配置界面如图 3-14 所示。

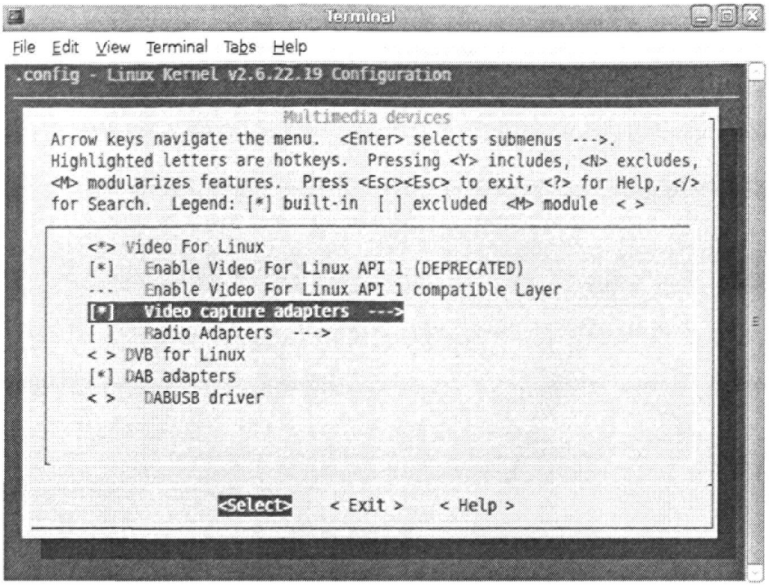

图 3-14 V4L 的内核配置界面

Fig3-14. V4L Configure Interface in Kernel

3.6.2 图像数据采集的实现

图像采集过程需要有摄像头驱动的支持，安装摄像头驱动，以便完成图像数据的采集过程。

3.6.2.1 摄像头驱动的安装

摄像头驱动是图像采集的基础，根据摄像头的芯片 Z0301 安装相应的驱动程序。本论文的摄像头驱动版本选用过的 gspca-20071224.tar 驱动版本，摄像头驱动的安装步骤如下：

1、解压 gspca-20071224.tar 后进入目录；

2、修改 Makefile 文件：

```
.....
KERNELDIR := /home/cuijie/linux-2.6.24
.....
CC = arm-linux-gcc
CROSS_COMPILE = arm-linux-
ARCH = arm
.....
```

3、编译驱动

#make

4、将生成的 gspca.ko 驱动文件下载到开发板上，手动安装或者系统自启动安装均可：

#insmod gspca.ko

摄像头驱动正确加载后，会检测摄像头设备的信息，返回摄像头的处理器及图像传感器等信息，如图 3-15 所示。处理器芯片为 ZC3XX，即 Z0301 系列处理芯片；图像传感器 MI0360；所获取图像最大为 640x480；驱动版本为 gspca-01.00.20。

```
[root@lyd /home]# insmod gspca.ko
/media/disk/cuijie/gspca-20071224/gspca_core.c: USB GSPCA camera found.(ZC3XX)
/media/disk/cuijie/gspca-20071224/gspca_core.c: [spca5xx_probe:4275] Camera type JPEG
/media/disk/cuijie/gspca-20071224/Vimicro/zc3xx.h: [zc3xx_config:679] Find Sensor MI0360. Chip r1
/media/disk/cuijie/gspca-20071224/gspca_core.c: [spca5xx_getcapability:1249] maxw 640 maxh 480 m0
usbcore: registered new interface driver gspca
/media/disk/cuijie/gspca-20071224/gspca_core.c: gspca driver 01.00.20 registered
```

图 3-15 摄像头信息

Fig3-15. Information of Camra

3.6.2.2 图像数据采集的实现

具备了摄像头驱动，Linux 通过 V4L 就可以实现对摄像头设备的控制和操作。进行图像采集前，首先需要在文件系统中建立/dev/video0 与/dev/v4l/vedio0 之间的链接，建立方法如下：

#mkdir /dev/v4l

#cd /dev/v4l

#mknod video0 c 81 0

#ln /dev/v4l/video0 /dev/video0

图像信息最终以 JPEG 格式的形式保存为图片文件。V4L 提供了相关的参数选择，JPEG 格式的设置方法如下：

#define VIDEO_PALETTE_JPEG 21

vd->mmap.format=VIDEO_PALETTE_JPEG;

图像采集开始先打开一个文件，然后利用 V4L 提供的函数，实现图像信息的设置及采集，再将数据写入文件。图像采集流程如图 3-16 所示。

3.7 图像传输模块

图像数据的传输是在覆盖网络中进行的，避免对传感网络造成太重的负担。

图片文件和视频信息的传输具有不同特点，视频信息更强调传输的实时性，下面分别介绍图片文件传输和视频传输的实现过程。

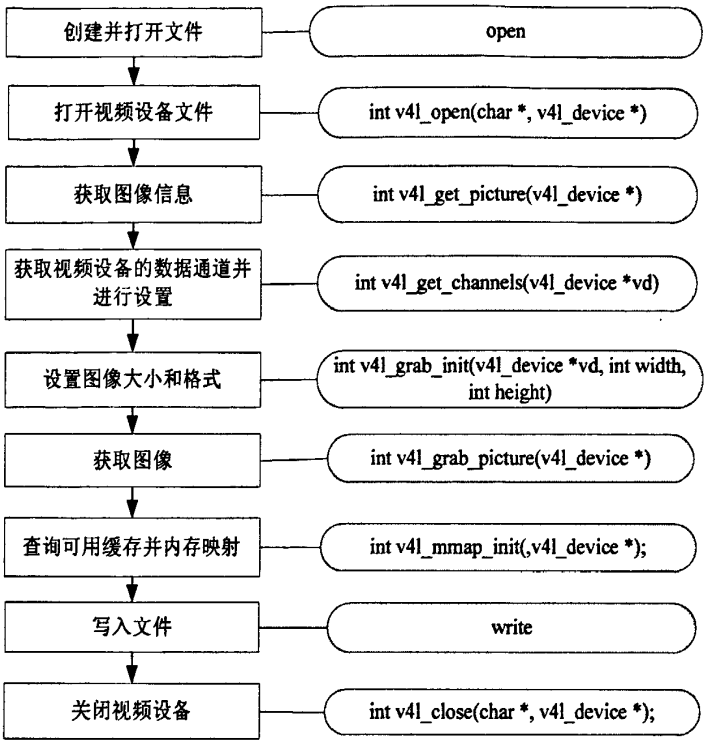

图 3-16 图像采集流程

Fig3-16. Procedure of Image Capturing

3.7.1 图片文件传输的实现

图片文件的传输采用 TCP 套接字。因为文件传输，对于数据包能否正确传输以及数据包到达目的地址的先后顺序的要求非常严格。鉴于 TCP 是通过三次握手实现连接的，另外还是用了超时重传和流量控制的机制。因而保证了数据包的发送和接受同步，提供了准确可靠的传输。

图片文件传输过程中，ARM 节点作为 TCP 连接的客户端，服务器可以作为服务器端， 用户则可以通过 JSP 界面观测到采集回来的数据。因此图片文件传输流程由发送和接收两部分组成，如图 3-17 所示。

图片文件发送实现的过程如下：

1、 创建 TCP 类型套接字，建立连接；

2、 发送文件名称；

3、 读取文件到 buf 中，然后发送，知道文件读取为空；

4、 关闭文件和套接字。

在服务器端对应的要有图像接收程序，实现的过程如下：

1、 创建 TCP 类型套接字，绑定本地地址，监听端口；

2、 通告发送端有空闲连接，并创建新套接字；

3、 接收文件名字符串和图像数据；

4、 以当前系统时间重新命名文件，用来记录图像接收到的时间；

5、 打开文件，将图像数据存入文件。

最终图像数据将以文件的形式存入服务器，用户可以查询到所有发送来的图像信息和对应的采集时间。

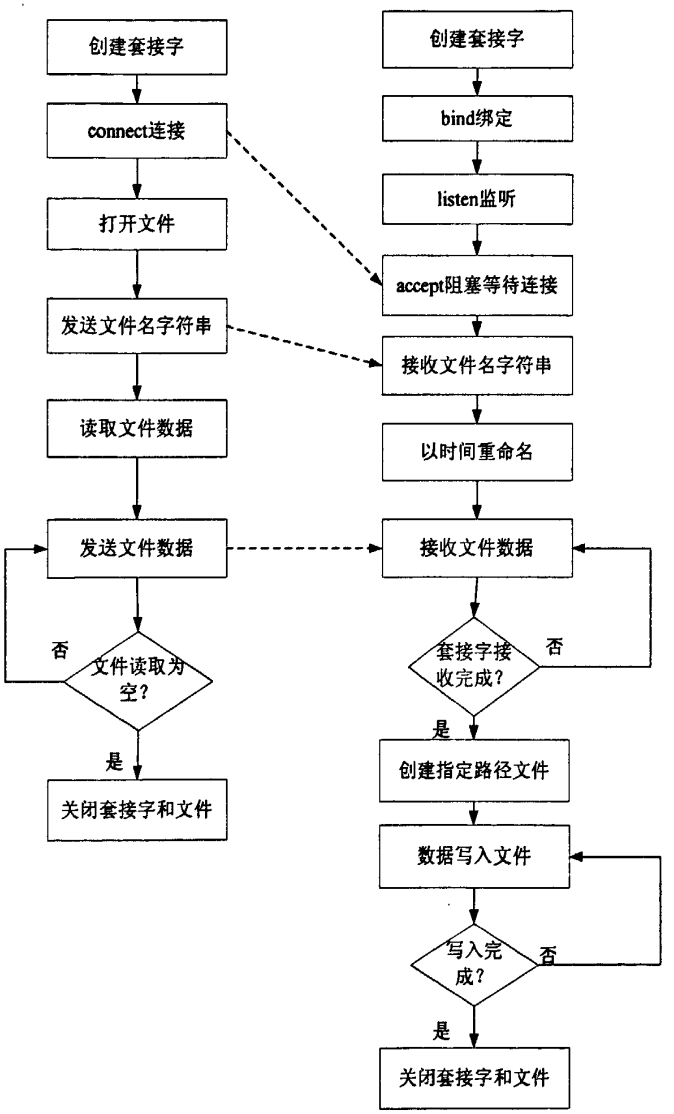

图 3-17 图片文件传输流程

Fig3-17. Flow of Image File Transmission

3.7.2 视频信息传输的实现

视频能够为用户提供更加直观的信息。直接利用开源的视频服务器端和客户端程序，做相应的修改可以实现实时视频传输功能模块。ARM 节点利用 Servfox 作为视频服务的客户端，服务器则使用 Spcaview 作为视频服务的服务器端。Servfox 和 Spcaview 是基于 V4L 和 TCP 的视频客户端与服务器端软件。

视频信息传输的实现过程如下：

1、移植 servfox

Servfox 是基于 V4L 和 TCP 的视频客户端软件，将 TCP 传输程序中的 IPv4 相关的套接字、地址等修改为 IPv6 的类型，以支持 IPv6 网络的视频传输。下载 servfox-R1_1_3.tar.gz 压缩包，解压后进入文件夹，拷贝 ARM 平台编译文件，然后编译：

```
#cp Makefile.arm Makefile
#make
```

将生成的可执行文件 servfox 下载到 ARM 开发板的相应文件夹下。

2、安装 spcaview

在服务器上，下载 SDL-1.2.14.tar.gz 和 spcaview-20061208.tar.gz 压缩包并解压，类似于 servfox 修改套接字传输部分，支持 IPv6。

安装 SDL (Simple DirectMedia Layer)开发包。SDL 是一个自由的跨平台的多媒体开发包，提供对视频服务的支持，进入 SDL-1.2.14 文件夹，编译安装：

```
#make
#make install
```

安装 spcaview，进入 spcaview-20061208 文件夹下，编译安装：

```
#make
#make install
```

3、运行客户端和服务器端程序

```
#servfox –d /dev/video0 –s 640x480 –w 3ffe:3240:8007:2004::2000:7070
#spcaview –w 3ffe:3240:8007:2004::****:7070
```

在服务器端就可以观测到 ARM 节点传回的实时视频信息。

3.8 多链路选择模块

异构无线传感器网络为数据传输提供了多种链路。多链路选择模块主要针对的是本地发送的数据以及需要转发的数据，分别在应用层和适配层对数据包进行

了判断和处理。多链路的选择还涉及到 IEEE802.15.4 和 IEEE802.11g 的网络地址的配置和绑定。

多链路选择模型如图 3-18 所示。ARM 节点的两种无线通信模块下的数据帧分别符合 IEEE802.15.4 协议标准和 IEEE802.11g 协议标准定义的数据帧格式。利用 Linux 内核协议栈，多链路选择模块实现了将接收到的 IEEE802.15.4 数据帧，经过多链路选择判断和处理之后，如果判断要转换到覆盖网络传输，则 ARM 节点可以将数据包转换为 IEEE802.11g 数据帧的格式发送出去。

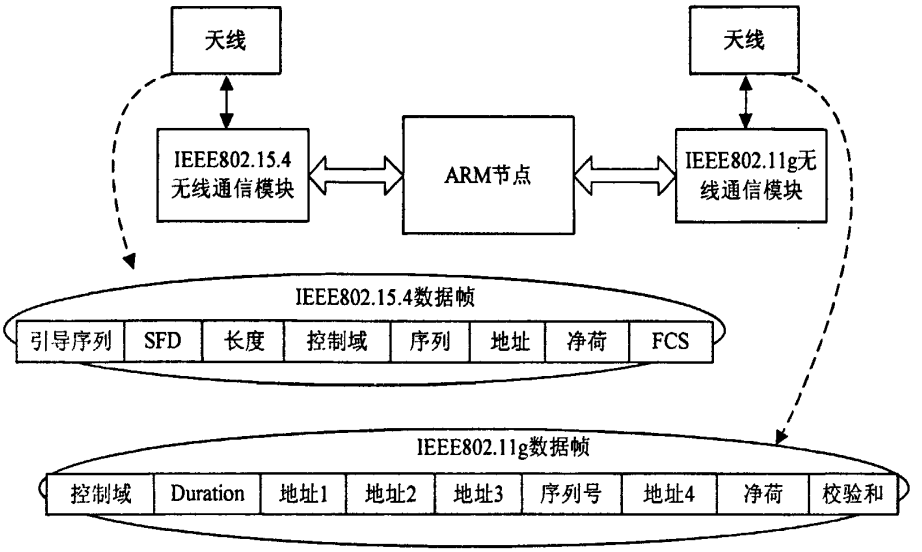

图 3-18 多链路选择模型

Fig3-18. Module of Multi-link Selection

3.8.1 应用层的多链路选择

应用层数据的多链路选择既可以在 ARM 节点上实现，也可以由单片机节点完成。应用层数据的处理在 ARM 节点和单片机节点上是相同的。通过在应用层设置报警阈值，可以时刻检测到当前数据的情况，在很多应用场景下，这种报警功能都是必不可少的。例如根据温度的迅速变化预警火灾的发生，尽可能得减少这类灾难造成的损失。

在应用层采用自定义简单类型传感数据在应用层的格式，appsensor 数据结构用于记录传感数据的相关信息，定义如下：

```
struct appsensor
{
  u8_t tag;  --标志位，值为 1 表示数据是接收来的，值为 2 说明数据是要发送出去的
```

```
    u8_t datat:6; --传感数据类型，例如，值为 3 表示温度数据
    u8_t appt:2; --数据包类型，值为 1 表示是控制信息，值为 2 表示为传感数据，值为 3 表
                示为警报数据
    u16_t appdata[4]; --存放数据数组
    u8_t hop; --跳数
    u8_t route[1];
};
```

应用层完成数据处理其他工作之后，会将采集的温度数值与规定的报警阈值进行比较。如果数值正常，则数据包类型字段值为 2，仍然在传感网络中传输；否则，修改数据包类型字段，改为 3。

对于 ARM 节点，适配层的多链路选择功能会截取数据包类型字段为 3 的包，进行网络地址更换的处理，处理后的数据包返回内核协议栈，内核将数据包交给 IEEE802.11g 驱动，该数据包将在覆盖网络中传输。

对于单片机节点，应用层处理完成后，数据将被直接发送，途径的 ARM 节点会对该数据进行判断处理。

选择在覆盖网络中传输警报数据，可以保证数据尽快的准确的返回服务器，以便监测者作出相应的措施。

3.8.2 适配层的多链路选择

在适配层发送数据的过程中，也会对数据包进行一个多链路的选择。适配层的多链路选择流程如图 3-19 所示。

执行发送数据队列的检测时，根据发送队列数据结构中的 flag 参数可以判断当前发送数据包的排队个数，如果检测到发送队列已满，则证明此时 IEEE802.15.4 无线链路繁忙，或出错，数据包的类型字段会修改为 3。由于适配层头部、IPv6 头部和 UDP 头部的长度都是固定的，所以可以从数据 buffer 中容易得判断出数据包类型字段。

ARM 节点在数据发送出去前的钩子点处会对该数据进行检测。检测结果为 3 的话，钩子将更换数据包的网络地址，返回协议栈由内核判断交给 IEEE802.11g 网络发送。

适配层的多链路选择主要针对的是两种情况：一是对于需要转发的数据包，如果源单片机节点已经将数据包的类型值置为 3，ARM 节点需要需要对这类型数据包选择 IEEE802.11g 链路传输；二是针对 IEEE802.15.4 的发送队列已满的情况，为了减轻 IEEE802.15.4 的网络负载，并减少丢包率，此时的数据包也会选择

IEEE802.11g 链路传输。

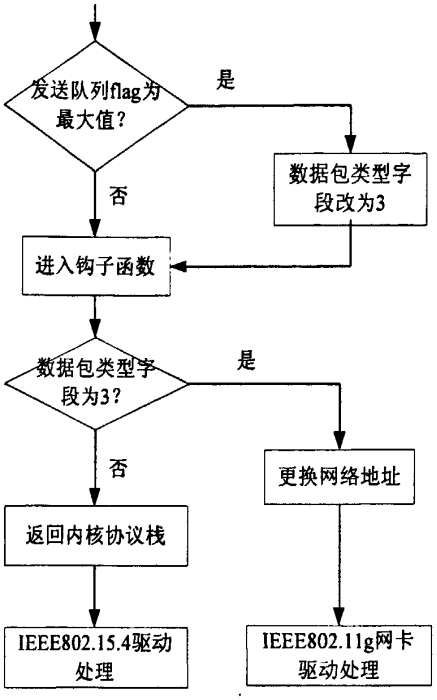

图 3-19 适配层多链路选择流程

Fig3-19. Flow of Multi-link Selection in Adapt Layer

3.8.3 网络地址绑定

ARM 节点具有两种无线通信接口，每个无线通信模块都有各自固定的 MAC 地址，所以 ARM 节点相应的也具有两个网络地址：IEEE802.15.4 和 IEEE802.11g 网络地址。网络采用的是 IPv6 协议，因此需要绑定 ARM 节点的两个 IPv6 地址，以方便服务器端识别节点标号，确定数据的源节点。

IPv6 网络地址提供了 128 位的地址空间，能够容纳多级的层次结构。一个 128 位 IPv6 地址由两部分组成：64 位子网标识符和 64 位接口标识符。

IEEE802.15.4 网络地址的配置如下：

➢ 链路本地地址的配置

配置 IPv6 链路本地地址过程中要保证其唯一性和合法性。其中，链路本地地址将作为同一链路内唯一标识网卡接口的地址。链路本地地址格式为 FE80::/64，前 64 位为链路本地地址的前缀，是固定值；后 64 位是接口 ID，而节点用唯一的 64 位的扩展地址 aExtendedAddress 确定。

➢ 全球单播地址的配置

节点在入网过程中，向父节点申请地址块，同时也申请到了网关的全球地址，

把网关地址的全局路由前缀，与接口标识符一起组成节点的全球单播地址，其中接口标识符仍然使用节点扩展地址 aExtendedAddress。

IEEE802.11g 的网络地址，前 64 位配置为同一网段，而后 64 位地址与 IEEE802.15.4 的后 64 位地址相同。接口标识符是基于 IEEE802.15.4 网络中唯一能够标识一个节点和其 IPv6 地址的标志，因此在配置 IEEE802.11g 网络的 IPv6 地址的时候，后 64 位也采用节点的扩展地址，与 IEEE802.15.4 网络地址一一对应，完成网络地址的绑定。论文的 IEEE802.15.4 地址格式为 3ffe:3240:8007:2004:xxxx:xxxx:xxxx:x，IEEE802.11g 的网络地址格式为 3ffe:3240:8007:1005:xxxx:xxxx:xxxx:x。

3.9 本章小结

这一章是本论文的核心部分。根据第二章提出的异构无线传感器网络，本章设计了异构无线传感器网络的多链路传输技术。具体介绍了初始化模块、任务调度模块、数据传输模块、数据采集模块、图像传输模块以及多链路选择模块的设计和实现过程，还说明了其中利用到的 Linux 内核相关的数据结构、函数、API 接口等。下一章将对多链路传输技术进行测试。

4 系统测试

本章在实验室的无线传感网络环境中对异构无线传感器网络多链路传输技术进行了测试。搭建测试环境，安装相应的测试工具，在此基础上分别建立了传感网络和覆盖网络，形成异构无线传感器网络，并利用红外-图像监控测试对多链路传输技术进行了验证，最后分析了测试结果并给出结论。

4.1 测试环境建立

4.1.1 测试环境

异构无线传感器网络多链路传输技术的测试是基于北京交通大学下一代互联网互联设备国家工程实验室无线传感网络环境的。多链路传输技术测试的网络拓扑如图 4-1 所示。

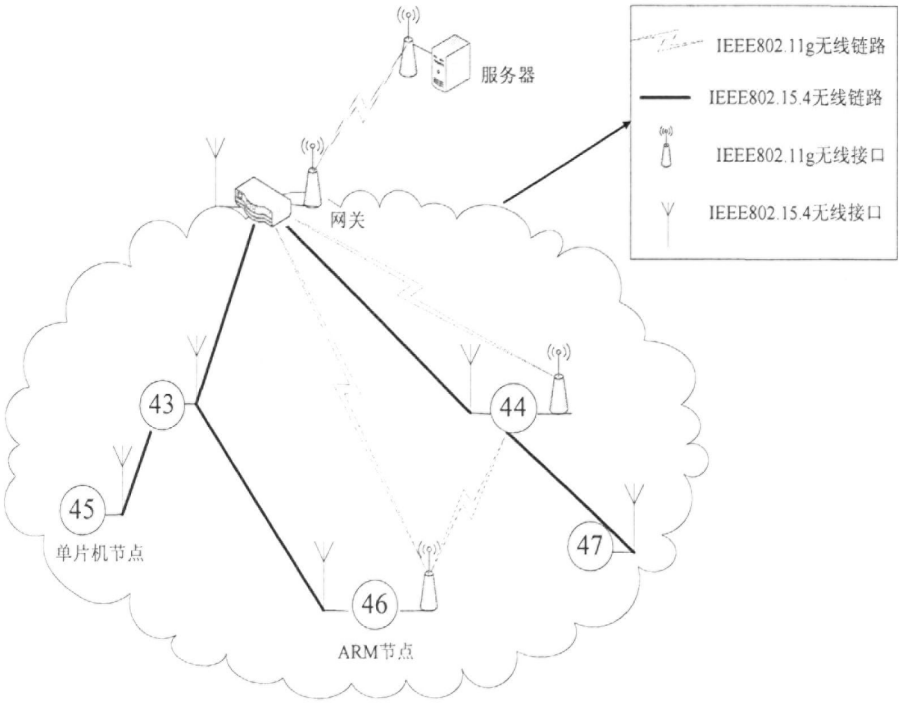

图 4-1 测试网络拓扑

Fig4-1. Topology of Test Network

如图 4-1 所示，测试网络包括一个网关，两个 ARM 节点和三个单片机节点，以及服务器。ARM 节点 44、46 和网关具有 IEEE802.15.4 和 IEEE802.11g 两种无

线接口，单片机节点 43、45 和 47 只具有 IEEE802.15.4 无线接口，服务器只具有 IEEE802.11g 接口。

4.1.2 相关工具

多链路传输技术的测试涉及到编译和抓包软件工具的使用，下面简要介绍相关工具的安装、配置和使用。

➢ 交叉编译工具链

针对嵌入式系统的用来交叉编译的工具包，本论文选用的交叉工具链为 arm-linux-gcc-3.4.1.tar.bz2，Linux 系统的交叉编译环境的建立过程如下：

1、下载 arm-linux-gcc-3.4.1.tar.bz2 工具包后解压，复制到任意目录下；

2、修改环境变量，在/etc/profile 最后添加

```
export PATH=$PATH:/home/cuijie/usr/local/arm/3.4.1/bin/:
```

交叉编译环境建立完成后，在宿主机上通过 arm-linux-gcc 编译生成可执行文件，下载到 ARM 开发板上即可以执行。

➢ Packet sniffer

Packet Sniffer 是一个用于 IEEE802.15.4 网络的数据包嗅探器。它能够完整地捕捉到所处无线网络中所有节点的上、下行数据包。论文选用版本为 Setup_Packet_Sniffer_1_2.exe，直接在 Windows XP 系统中安装。

➢ Wireshark

Wireshark 是一个网络封包分析软件。本文主要利用 Wireshark 抓取异构无线传感器网络网关向服务器端发送的数据包进行分析。Linux 系统下的安装方法如下：

```
#apt-get install wireshark
```

➢ Minicom

Minicom 是串口显示工具，用于 ARM 节点的运行信息的显示。Linux 系统下安装及配置过程如下：

1、安装 minicom：

```
#apt-get install miniocm
```

2、配置 minicom，先进入配置界面：

```
#minicom -s
```

3、进入 Serial Port setup 选项进行配置，各项参数设置如图 4-2 所示。

Minicom 安装和配置完成后，用串口线将宿主机与 ARM 开发板相连，ARM 节点的运行信息就可以在 minicom 中打印显示。

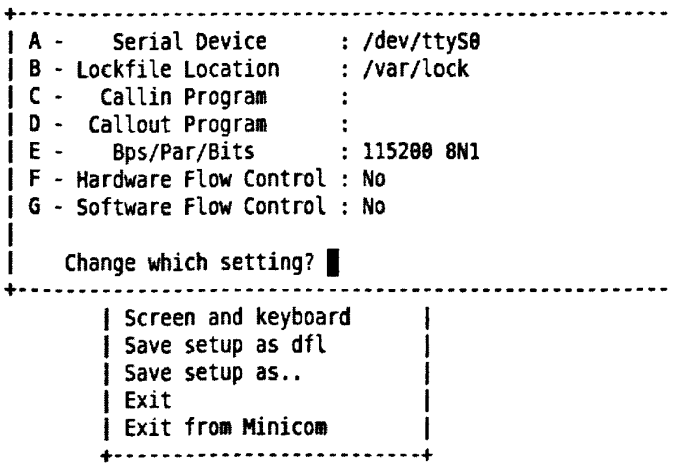

图 4-2 minicom 的配置

Fig4-2.Configuration of Minicom

4.2 多链路传输技术的验证

多链路传输技术的验证分为传感网络建立、覆盖网络建立和多链路传输技术测试三个方面。

4.2.1 传感网络的建立

单片机节点和 ARM 节点启动后会与网关或父节点之间进行一些信息交互，以完成节点的入网过程。入网完成后，数据可以在传感网络中传输，完成传感网络的建立。

保证服务器和网关正常运行后，按照节点的标号顺序打开节点电源，节点会依次入网。ARM 节点的入网是基于 IEEE802.15.4MAC 驱动的。ARM 节点 44 直接向网关发送信标请求（Beacon Ruquest）、入网请求（Association Request）、数据请求（Data Request），网关回复相应的 ACK，完成入网。ARM 节点 46 则通过父节点 43 入网。

节点完成入网之后，节点开始定时循环发送传感数据。服务器接收到数据后存入数据库，并根据数据包内的信息，绘制出数据包的传输路径在界面上显示。传感网络的数据传输如图 4-3 所示。

实验表明 ARM 节点与单片机节点完成了传感网络的建立过程。ARM 节点基于 Linux 操作系统实现了单片机节点的基本功能。在传感网络中，ARM 能够与单片机节点建立路由，并实现数据包的发送和转发。

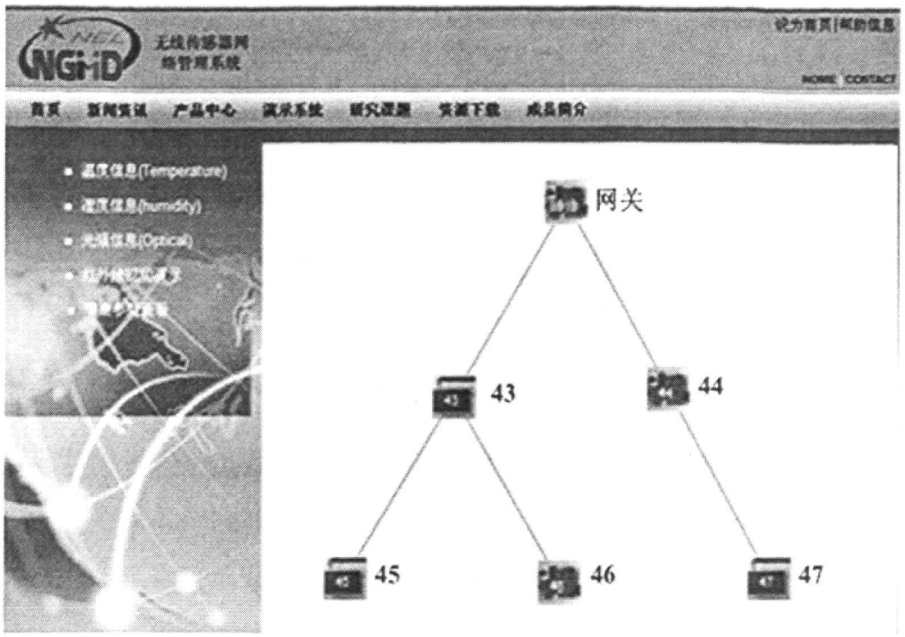

图 4-3 传感网络的数据传输

Fig4-3. Data Transmission of Sensor Network

4.2.2 覆盖网络的建立

在节点 44、节点 46、网关和服务器上安装配置无线网卡，安装和配置方法详见 2.2.2.2 小节和 3.2.2.2 小节。无线网卡配置完成后，添加 AODV 路由协议，建立覆盖网络。

下载 aodv-uu-9.0.5.tar.gz 源码包，解压后，分别用 gcc 和 arn-linux-gcc 编译，将交叉编译后的驱动文件和可执行文件下载到节点 44、节点 46 和网关。安装驱动并运行可执行文件，运行 AODV 路由协议。

```
#insmod kaodv.ko
#./aodvd
```

考虑到服务器与节点之间的距离一般超过了节点 44 和节点 46 的通信范围，并且为了测试 AODV 路由协议，因此先通过命令禁止 ARM 节点与服务器端的通信，如图 4-4 所示。

服务器的 MAC 地址为 00:24:8c:6b:98:99，通过设置，ARM 节点将丢弃所有来自服务器的数据包。命令如下：

```
#iptables –A INPUT –p ALL –m mac –mac-source 00:24:8c:6b:98:99 –I DROP
```

发送 ping 命令包测试 AODV 路由协议，在服务器端输入 ping 命令：

```
#ping6 3ffe:3240:8007:1005:4619:6392:4700:0
```

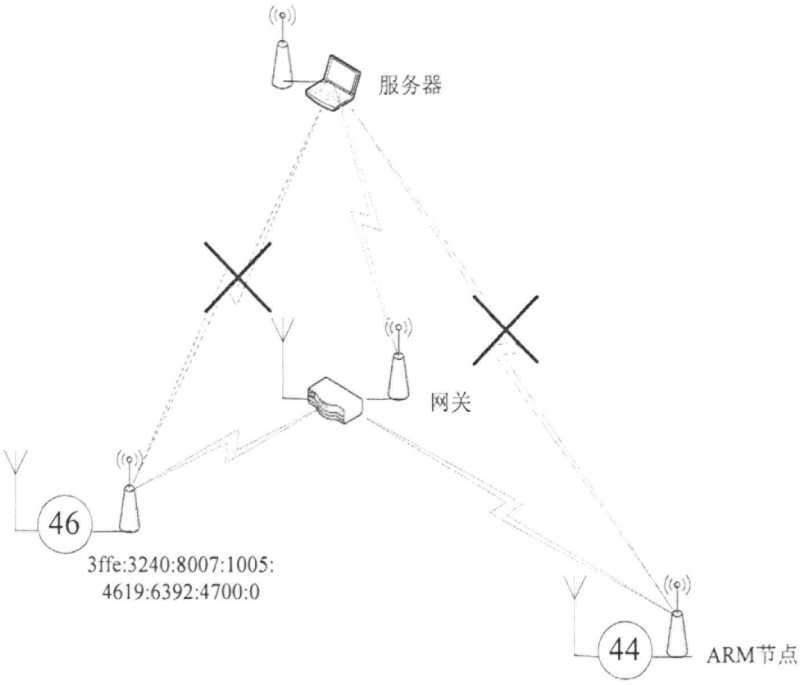

图 4-4 覆盖网络

Fig4-4. Overlay Network

服务器无法与节点 46 直接通信，因此通过网关与节点 46 建立起了 AODV 路由，ping 包测试如图 4-5 所示。

```
debian:~# ping6 3ffe:3240:8007:1005:4619:6392:4700:0
PING 3ffe:3240:8007:1005:4619:6392:4700:0(3ffe:3240:8007:1005:4619:6392:4700:0) 56 data bytes
64 bytes from 3ffe:3240:8007:1005:4619:6392:4700:0: icmp_seq=1 ttl=64 time=4.87 ms
64 bytes from 3ffe:3240:8007:1005:4619:6392:4700:0: icmp_seq=2 ttl=64 time=6.98 ms
64 bytes from 3ffe:3240:8007:1005:4619:6392:4700:0: icmp_seq=3 ttl=64 time=5.78 ms
64 bytes from 3ffe:3240:8007:1005:4619:6392:4700:0: icmp_seq=4 ttl=64 time=6.25 ms
64 bytes from 3ffe:3240:8007:1005:4619:6392:4700:0: icmp_seq=5 ttl=64 time=5.40 ms
64 bytes from 3ffe:3240:8007:1005:4619:6392:4700:0: icmp_seq=6 ttl=64 time=5.58 ms
64 bytes from 3ffe:3240:8007:1005:4619:6392:4700:0: icmp_seq=7 ttl=64 time=5.47 ms
64 bytes from 3ffe:3240:8007:1005:4619:6392:4700:0: icmp_seq=8 ttl=64 time=4.64 ms
64 bytes from 3ffe:3240:8007:1005:4619:6392:4700:0: icmp_seq=9 ttl=64 time=6.73 ms
64 bytes from 3ffe:3240:8007:1005:4619:6392:4700:0: icmp_seq=10 ttl=64 time=6.85 ms
```

图 4-5 ping 包测试

Fig4-5.Test of Ping Packets

4.2.3 红外-图像监控测试

通过红外-图像监控测试可以对多链路传输技术进行测试。红外监控测试环境如图 4-6 所示，包括红外单片机节点、ARM 节点、网关和服务器。其中节点 47 为单片机节点，在上面配备红外传感模块；节点 44 为 ARM 节点，添加摄像头作为图像传感模块。

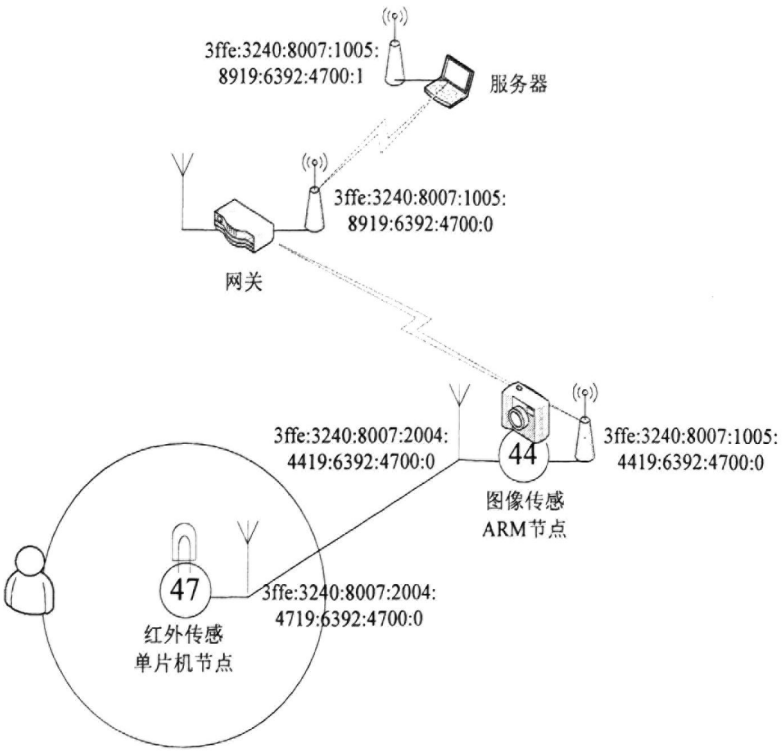

图 4-6 多链路测试环境

Fig4-6. Test of Multi-link

4.2.3.1 多链路选择测试

当人体进入红外传感器感应范围内时，基于红外传感单片机节点 47 由上升沿触发，发送红外传感数据包，该数据包默认是普通的传感数据，因此数据包类型字段值为 2，抓取的普通红外数据包如图 4-7 所示。普通红外数据包经过 ARM 节点后，ARM 节点多链路判断数据包在传感网络中传输，直接转发给网关，网关再将数据包发送到服务器。

Time (us)	Length	Frame control field								Sequence number	Dest. PAN	Dest. Address	Source Address
+14598377		Type	Sec	Pnd	Ack req	Intra PAN							
=52034857	86	DATA	0	0	1	1				0x88	0x2420	0x0000004762931944	0x0000004762931989

MAC payload																										LQI	FCS
04	00	60	00	00	00	00	15	11	FF	3F	FE	32	40	80	07	12	09	49	19	63	92	47	00	00	00		
12	09	19	89	95	22	47	00	00	00	04	02	0A	1C	00	15	38	1E	02	49	11	00	22	00	33	00	0	OK

图 4-7 普通红外传感数据包

Fig4-7.Common Infra-red Packets

服务器收到的普通红外数据包如图 4-8 所示。源地址为红外传感单片机节点的 IEEE802.15.4 网络地址 3ffe:3240:8007:2004:4719:6392:4700:0。

图 4-8 服务器接收的普通红外数据包

Fig4-8. Common Infra-red Packets Received by Server

为方便测试应用层数据的多链路选择功能，红外传感单片机节点的警报阈值设为 0。当继续有人进入红外感应区域时，再次触发节点 47 发送红外传感数据。与上次缓存的数据差运算，并与阈值进行比较后，节点判断已经达到警报阈值，因此将数据包类型字段改为 3。紧急红外传感数据包如图 4-9 所示。

Time (us)	Length	Frame control field						Sequence number	Dest. PAN	Dest. Address	Source Address
+3704492		Type	Sec	Pnd	Ack	req	Intra PAN				
+56045349	86	DATA	0	0	1		1	0x88	0x2420	0x0000004762931944	0x0000004762931989

MAC payload	LQI	FCS
04 00 60 00 00 00 00 15 11 FF 3F FE 32 40 80 07 12 09 49 19 63 92 47 00 00 00 3F FE 32 40 80 07		
12 09 19 89 95 22 47 00 00 00 04 02 0A 1C 00 15 38 1E 03 49 11 00 22 00 33 00 44 00 01 49 19	0	OK

图 4-9 紧急红外传感数据包

Fig4-9.Urgent Infra-red Packets

ARM 节点收到紧急红外传输数据包后，数据传输模块完成数据的转发，数据包发送过程中，进入 ARM 节点的适配层多链路选择。由于判断数据包类型字段为 3，ARM 节点将数据包的源地址更换为源节点（即红外传感单片机节点）的 IEEE802.11g 地址，并通过覆盖网络将数据返回服务器。因此，服务器端接收到的紧急红外数据包如图 4-10 所示。与图 4-8 对比可以看出，紧急红外传感数据包的源地址已经变为 3ffe:3240:8007:1005:6392:4700:0。表明 ARM 节点已经对紧急红外传感数据包进行了多链路选择的处理。

图 4-10 服务器接收的紧急红外数据包

Fig4-10.Urgent Infra-red Packets Received by Server

4.2.3.2 图像数据采集和传输测试

ARM 节点 44 接收到红外数据包后，将数据包由内核空间送往用户空间，用户空间的图像采集功能启动，采集完的图像信息通过网关返回到服务器。服务器端显示接收到的图像信息如图 4-11 所示。

图 4-11 图像数据

Fig4-11.Image Data

4.3 实验结果及分析

本论文在测试环境下，搭建了异构无线传感器网络，分别建立了传感网络和覆盖网络，并利用红外-图像监控对多链路传输技术进行了测试。

- 传感网络建立

ARM 节点启动后通过抓取数据包显示 ARM 节点能够正常地加入传感网络，与单片机节点之间进行通信。由服务器端的数据接收显示可以看出 ARM 节点也能够在传感网络中进行普通的数据传输。

- 覆盖网络建立

ARM 节点、网关和服务器上配置了无线网卡，并安装 AODV 协议，通过 ping 包测试，验证了 ARM 节点、网关与服务器之间能够在 AODV 建立的路由上进行通信。从而验证了覆盖网络建立成功。

- 红外-图像监控测试

红外单片机节点第一次触发后发送普通的红外数据包，ARM 节点接收到该数据包后，多链路选择判断的结果为普通传感类型数据，因此仍然在传感网络中将数据返回服务器。而当红外传感单片机节点第二次触发发送的红外数据包，由应用层的多链路选择功能设置该数据包为紧急数据包，ARM 节点接收到紧急数据包后，根据多链路选择判断数据包需要转到覆盖网络进行传输，因此 ARM 节点对数据包进行地址变换等处理，通过无线网卡将数据发送到网关，网关又将红外数据包转发给服务器。对比红外单片机节点发送的数据包与服务器接收的数据包的地址可以验证 ARM 节点实现了多链路选择的功能。

此外，ARM 节点接收到红外数据包后，启动了图像采集模块，图像信息也通过覆盖网络传回了服务器，在服务器端的界面显示，这部分工作在文献[30]中已经实现并验证。

红外-图像监控测试验证了 ARM 节点实现了不同的实际情况对数据进行多链路选择传输。

4.4 本章小结

本章异构无线传感器网络的多链路传输技术进行了测试。首先介绍了总体测试环境和相关测试工具以及测试工具的安装。然后分别搭建了传感网络和覆盖网络，完成异构无线传感器网络的建立过程，并进行了多链路传输的测试。最后对实验结果进行了分析。

5 总结与展望

随着异构无线传感器网络的研究和应用，多链路传输技术的研究也更加重要。本论文根据设计的异构无线传感器网络，对多链路传输技术进行了设计、实现和测试，主要完成的工作如下：

1、分析并研究了异构无线传感器网络现状及存在的问题。针对单片机节点和ARM 节点设计了异构无线传感器网络的结构。

2、设计并实现异构无线传感器网络下的多链路传输系统。实现 ARM 节点上软件系统平台的移植，在系统平台上划分初始化模块、任务调度模块、数据传输模块、数据采集模块、图像传输模块以及多链路选择模块。详细设计并实现了各功能模块。

3、搭建测试环境，对异构无线传感网络的连通性和多链路传输进行了测试，并分析了实验结果。

本论文设计并实现了异构无线传感网络的多链路传输系统。随着研究的深入，仍然有些问题需要进一步研究，因此下一步的工作主要包括以下几个方面：

1、多链路选择的算法。本论文仅采用了简单的多链路选择机制。目前的异构无线网络采用了模糊层次分析法作为多链路选择的理论依据和算法设计，也可以将这种模糊层次分析法应用到异构无线传感器网络中。可以考虑信号强度、无线链路质量、网络的拥塞度、能量消耗估计等方面的参数。

2、网络部署策略。选用合适数量的单片节点和 ARM 节点，并采取一定的部署策略搭建网络，以达到最优的网络性能。

3、抗干扰机制。IEEE802.15.4 和 IEEE80.11g 的工作频带都是 2.4GHz，会有相互干扰的影响。需要研究抗干扰机制减小干扰造成的影响。

参考文献

[1] 孙利民, 李建中编著.无线传感器网络[M].第一版.清华大学出版社.2005.4.10.

[2] Jennifer Yick, Biswanath Mukherjee, Dipak Ghosal, Wireless sensor network survey[J]. Computer Networks, Vol.52. 2008. Page(s): 2292–2330.

[3] Krishnamurthy Lakshman, Connect the Physical World to Information Technology: Industrial application for sensor networks[EB/OL]. Advanced sensing Technologies, Spring 2004 seminar series, http://asia. Stanford. edu/events/Spring04/sensors.html.

[4] Lakshman Krishnamurthy, Sensor Networks: Promise and Reality, incited presentation, Sensors Expo and Conference, Detroit, ML June 9, 2004.

[5] Estrin D,Culler D,Pister K, et al. Connecting the Physical World with Pervasive Networks[J],IEEE Pervasive Computing,2002, 1(1): 59 – 69.

[6] Mark Yarvis, Nandakichore Kunshalnagar, Harkirat Singh, Anand Rangarajan, York Liu, and Suresh Singh, Exploiting Heterogeneity in Sensor Nerworks[C]. Proceedings of the IEEE International Conference on Computer Communication, Miami, FL, March 2005, (2):878-890.

[7] J.Hill, R.Szewezyk, A. Woo, S. Hollar, D. Culler, and K. Pister, System Architecture Directions for Networked Sensors, Proc. of ASPLOS 2000, Cambridge, MA, 2000.

[8] Ardizzone E, Cascia ML, Re GL, Ortolani M. An integrated architecture for surveillance and monitoring in an archaeological site Proc. ACM Int'l Workshop Video Surveillance and Sensor Networks, ACM Press, 2005, pp. 79-85.

[9] Akyildiz, I.F., Melodia,T.Chowdury, K.R., Wireless multimedia sensor networks: A survey[J]. IEEE WIRELESS COMMINICATION, Vol.56.2007.Page(s):32-39.

[10]Grey Hawk. Cooperative Engagement Capability(CEC)[C]. U. S. Washington D. C.: network centric air defense, 2001 / 6.

[11] CodeBlue: Wireless Sensor Networks for Medical Care, Division of Engineering and Applied Sciences, Harvard University, http://www.eecs.harvard.edu/~mdw/proj/codeblue/.

[12]谢雨珊，美国运用 WSN 于健康医疗照护之实际案例分析，http://www.itis.org.tw/rptDetailFreeEPaper.screen?rptidno=764343774, 2009.

[13] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, Wireless Sensor Networks for Habitat Mornitoring, Intl.Workshop on Wireless Networks and Applications, Atlanta, GA,2002.

[14] MA Fu-chang, FENGDao-xun, ZHANG Ying-mei, Research of wireless network based on technique of ZigBee and GPRS, Automation & Instrumentation, 2008-03.

[15] Li Qing, Qingxin Zhu, Mingwen Wang, Design of a distributed energy-efficient clustering algorithm for heterogeneous wireless sensor networks, Computer Communications, August 2006, Page(s): 2230-2237.

[16] R.Kumar, V.Tsiatsis, and M.B.Srivastava, Computation Hierarchy for In-Network Processing, Proc. of the 2nd Intl.Workshop on Wireless Networks and Applicatons, San Diego, CA, 2003.

[17] 杨水清, 张剑, 施云飞, ARM 嵌入式 Linux 系统开发技术详解[M], 第一版.电子工业出版社, 2008.11.1.

[18] Atmega128 Datasheet, http://www.datasheetarchive.com.

[19] Samsung S3C2440 Datasheet, http://www.datasheetarchive.com/S3C2440-datasheet.htm.

[20] CC2420 Datasheet, http://www.datasheetarchive.com.

[21] AUSU WL-167G, www.asus.com.cn.

[22] Vimicro Corporation, Datasheet ZC0301 VGA USB PC Camera Controller, http://www.chinaicmart.com/part/z/91.html.

[23] J.Elson,S.Bien,N.Busek,V.Bychkovskiy,A.Cerpa,D.Ganesan,L.Girod,B.Greenstein,T.Schoellhammer,T.Stathopoulos, and D.Estrin .EmStar: An Environment for Developing Wireless Embedded Systems Software. Center for Embedded Networked Sensing, University of California, Los Angeles. March 24,2003 :1–2.

[24] Tanenbaum A S , 计算机网络[M] . 潘爱民译, 清华大学出版社, 2004.

[25] IEEE Std 802.15.4-2003 IEEE standard for information technology: Telecommunications and information exchange between systems: Local and metropolitan area networks specific requirements: Part 15.4 Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)[S].2003.

[26] IEEE 802.11g-2003 IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications—Amendment 4: Further Higher-Speed Physical Layer Extension in the 2.4 GHz Band.

[27]C. Perkins, E. Belding-Royer, S. Das. RFC3561. Ad hoc On-Demand Distance Vector (AODV) Routing [S]. http://moment.cs.ucsb.edu/pub/rfc3561.txt. 2003.7.

[28] Love,R. 著, Linux 内核设计与实现, 陈莉君等译, 第二版 [M] ,机械工业出版社, 2006.

[29] 王滔, 季晓勇, 在嵌入式 Linux 平台上使用 USB 摄像头, 微计算机应用, 27(1), 2006, Page(s): 52-54.

[30] 崔捷, 周华春, 高德云, 郑涛, 基于 ARM 的 WSN 图像监控系统的设计与实现[J], 现代电子技术, 2010，33(14):162-166.

作者简历

女，山西人，研究方向：计算机网络通信、无线传感器网络、嵌入式操作系统

教育经历：

2008.9-2011.1	北京交通大学	通信与信息系统	工学硕士
2005.9-2008.6	中国农业大学	理科实验班	工学学士

科研项目：

2008.11-2009.12：国家 863 计划"IPv6 低速无线个域网体系结构与组网技术"（项目编号: 2007AA01Z241）

2009.01-至今：国家发展改革委项目"下一代互联网技术实现智能建筑节能的应用示范"（项目编号: No.CNGI-09-03-05）

2010.01-至今：国家自然科学基金项目"面向紧急事件的无线传感器网络信息可靠传输理论与关键技术"（项目编号: No. 60972010）

学位论文数据集

表 1.1： 数据集页

<table>
<tr><td>关键词*</td><td>密级*</td><td>中图分类号*</td><td>UDC</td><td>论文资助</td></tr>
<tr><td>异构无线传感器网络；多链路传输技术；IEEE802.15.4；IEEE802.11g</td><td>公开</td><td>TP212.09；TN915.04</td><td></td><td>CNGI“下一代互联网技术实现智能建筑节能的应用示范”项目</td></tr>
<tr><td colspan="2">学位授予单位名称*</td><td>学位授予单位代码*</td><td>学位类别*</td><td>学位级别*</td></tr>
<tr><td colspan="2">北京交通大学</td><td>10004</td><td>工学</td><td>硕士</td></tr>
<tr><td colspan="2">论文题名*</td><td colspan="2">并列题名</td><td>论文语种*</td></tr>
<tr><td colspan="2">异构无线传感器网络多链路传输技术的设计与实现</td><td colspan="2"></td><td>中文</td></tr>
<tr><td>作者姓名*</td><td colspan="2">崔捷</td><td>学号*</td><td>08120222</td></tr>
<tr><td colspan="2">培养单位名称*</td><td>培养单位代码*</td><td>培养单位地址</td><td>邮编</td></tr>
<tr><td colspan="2">北京交通大学</td><td>10004</td><td>北京市海淀区西直门外上园村 3 号</td><td>100044</td></tr>
<tr><td colspan="2">学科专业*</td><td>研究方向*</td><td>学制*</td><td>学位授予年*</td></tr>
<tr><td colspan="2">通信与信息系统</td><td>无线传感器网络</td><td>两年半</td><td>2011</td></tr>
<tr><td>论文提交日期*</td><td colspan="4">2010.11.22</td></tr>
<tr><td>导师姓名*</td><td colspan="2">周华春</td><td>职称*</td><td>教授</td></tr>
<tr><td>评阅人</td><td colspan="2">答辩委员会主席*</td><td colspan="2">答辩委员会成员</td></tr>
<tr><td></td><td colspan="2">秦雅娟</td><td colspan="2">高德云、罗洪斌</td></tr>
<tr><td colspan="5">电子版论文提交格式 文本（√） 图像（ ） 视频（ ） 音频（ ） 多媒体（ ） 其他（ ）
推荐格式：application/msword；application/pdf</td></tr>
<tr><td colspan="2">电子版论文出版（发布）者</td><td colspan="2">电子版论文出版（发布）地</td><td>权限声明</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td><td></td></tr>
<tr><td>论文总页数*</td><td colspan="4">76</td></tr>
<tr><td colspan="5">共 33 项，其中带*为必填数据，为 22 项。</td></tr>
</table>